

PAC LEARNING A DECISION TREE WITH PRUNING

By

HYUNSOO KIM

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1992

Copyright 1992

by

Hyunsoo Kim

To my parents, my sisters and brother,
and my wife who made this possible

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere appreciation to Dr. Gary J. Koehler, chairman of my supervisory committee and my department. My research here was guided by his invaluable advice and continuous encouragement. I am also deeply grateful to his kindness in every aspect of my life in the United States.

I am grateful to Dr. Harold P. Benson for his ever clear and organized teaching in courses which trained me for theoretical research.

I am indebted to Dr. Selcuk Erenguc for his considerate guidance and sincere help in my graduate studies.

I also would like to thank to Dr. William Messier for his kind advice on my research, and to Dr. Mark Pendergast for serving on this committee.

Finally I would like to thank all my family members for their support and encouragement. My special thanks go to my wife, Yeongsil, who is always with me and thinks and feels for me more than I do. My final thanks are directed to my daughter, Seoyoung, who always gives me a special motivation for study.

TABLE OF CONTENTS

	<u>page</u>
ACKNOWLEDGMENTS	iv
ABSTRACT	vii
CHAPTERS	
1 INTRODUCTION	1
1.1 Motivation of Research	1
1.2 Problem Statement	3
2 THEORETICAL BACKGROUND AND RELATED RESEARCH	8
2.1 Learning Theory	8
2.2 Decision Tree Induction Methods	21
2.3 Focus of Research	44
3 PAC LEARNING A DECISION TREE WITH PRUNING	46
3.1 Introduction	46
3.2 Binary Decision Trees	47
3.3 Pruning a Consistent Decision Tree to a Desired Rank	52
3.4 Determining the Pruning Error	63
3.5 Sample Size Sufficient for PAC Identification	78
3.6 Other Pruning Rules	87
3.7 Chapter Summary	93
4 THE ACCURACY OF A PRUNED DECISION TREE	95
4.1 Introduction	95
4.2 Estimations of Error of a Pruned Decision Tree	96
4.3 An Application	113
5 AN INVESTIGATION ON THE CONDITIONS OF PRUNING ..	115
5.1 Introduction	115
5.2 Fundamental Situation of Pruning	116
5.3 A Bayesian Analysis on the Conditions Where Pruning is Useful	118

5.4 A Generalization on the Conditions Where Pruning is Useful	124
6 SUMMARY AND FUTURE RESEARCH	146
REFERENCES	149
BIOGRAPHICAL SKETCH	156

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

PAC LEARNING A DECISION TREE WITH PRUNING

By

HYUNSOO KIM

May 1992

Chairman: Dr. Gary J. Koehler
Major Department: Decision and Information Sciences

This dissertation investigates various theoretical effects of pruning a decision tree. Empirical results have shown that pruning can improve the accuracy of an induced decision tree. Pruning also leads to more concise rules. First we provide a pruning algorithm based on the rank of a decision tree. A bound on the error due to pruning by the rank of a decision tree is determined under the assumptions of an equally likely distribution over the instance space and a deterministic tree labelling rule. This bound is then used with recent results in learning theory to determine a sample size sufficient for Probably Approximately Correct (PAC) identification of decision trees with pruning. We also discuss other pruning rules and their effects on the error due to pruning. With a nondeterministic tree labelling rule, we show that the upperbound of the average pruning error is less than or equal to 0.5 under an equally likely distribution.

In a realistic learning environment it is often not possible to obtain a large enough sample to guarantee PAC learning. For those cases, we provide several methods for a posterior evaluation of the accuracy of a pruned decision tree. We give a method which estimates a lower bound for the worst possible confidence factor, δ , by using a Beta prior. Also, we give a more detailed view of the meaning of this lower bound, and suggest a way to improve this lower bound.

Finally we develop conditions under which pruning is necessary for better prediction accuracy as well as for concept simplification. We give an analysis of the reason why pruning is necessary in realistic learning situations.

We generalize a previous result for larger training sets. A Bayesian analysis shows that the average prediction accuracy of the pruned tree increases, and the effect of description noise becomes stronger as the size of the training set increases. For very large training sets, the pruned tree has the prediction accuracy equal to that of the unpruned tree.

CHAPTER 1 INTRODUCTION

1.1 Motivation of Research

Learning is a general term denoting the way in which people or computers increase their knowledge or improve their skills (Cohen and Feigenbaum, 1982). There are several views of learning. One comprehensive view is that "learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population" (Simon 1983). Many expert systems treat learning as the acquisition of explicit knowledge. There are also other views such that learning is skill acquisition or that learning is theory formation or discovery.

Human learning is a long slow process. Overcoming such inefficiencies of human learning provides primary motivation for research in machine learning (Simon 1983). There are more compelling reasons for machine learning in the business environment. Expert systems are indispensable to modern business. Learning has been an important and effective method in acquiring knowledge for expert systems. Knowledge acquisition is the major bottleneck in expert systems development since it is difficult to elicit an expert's

knowledge by reconstruction methods such as interviewing an expert (Musen 1989). The more an expert knows, the less able is he to articulate that knowledge. This phenomenon is known as "the paradox of expertise" (Johnson 1983). Since experts often find it hard to articulate their expertise, many researchers are trying to develop alternative knowledge acquisition methods such as machine learning.

As we develop efficient automatic reasoning procedures which are applicable for knowledge acquisition, it is possible that the bottleneck of expert systems will be solved and expert systems will become more practical.

Expert systems have been used in business applications as diverse as production, marketing, finance, accounting, personnel and strategic management. Several benefits of expert systems are as follows (Parker 1989):

- .They preserve knowledge that might be lost through retirement, resignation or death of an acknowledged company expert.

- .They put information into an active form, so it can be summoned almost as one might summon a real-life expert.

- .They can assist novices to think like experienced professionals.

- .They are not subject to such human failures as fatigue, being too busy or being subject to emotions.

These benefits can lead to lower costs, better service, higher sales, and possibly, significant competitive advantages.

As expert systems are used in everyday business, our business environment will have a wholly different shape with flexible labor force utilization and increased productivity.

There is also a down-side. The office of the future may become the factory of the past (Garson 1988). As tasks that require creativity and expertise are performed by the computerized system in a highly automated and mechanized way, many professional people may be de-skilled and lose the joy they feel in planning and carrying out tasks.

1.2 Problem Statement

Cohen and Feigenbaum (1982) divide the topic of learning into four areas: rote learning, learning by being told (advice taking), learning from examples (induction), and learning by analogy. A brief description of the four learning situations follows. Suppose that a learning system is embedded in an environment of interest and a knowledge base is used by the performance element. Here the knowledge base is a collection of knowledge and the performance element is a system performing tasks by using the knowledge base.

1. Rote learning: The environment supplies knowledge in a form that can be used directly by the performance element.

The learning system just needs to memorize the knowledge for later use. Though rote learning is a rudimentary type of learning, this is widely used in daily human life.

2. Learning by being told: Here the environment gives vague, general purpose knowledge or advice. A learning system must transform this high-level knowledge into a form that can be used readily by the performance element. Davis's (1982) TEIRESIAS is an example of this type of learning.

3. Learning from examples: In learning from examples, examples are given to the learning system. The system generalizes these examples to find higher level rules that can be used by the performance element. This type of learning has a long history under the name of "induction" and is a powerful method of acquiring knowledge.

4. Learning by analogy: If a system has available to it a knowledge base for a related performance task, it may be able to improve its own performance by recognizing analogies and transferring the relevant knowledge from the other knowledge base.

Some researchers add more types of learning to the above four types of learning. For instance, Shaw et al. (1990) list two more types of learning, learning by competition, and learning from observation and discovery.

Learning situation can also be categorized by settings where learning takes place. There are two typical settings, one is "supervised" learning where a teacher is present and

can tell anything about the training set, and the other is "unsupervised" learning where a learning system has no instructor.

Inductive learning is generally considered synonymous with learning from examples. However, Angluin and Smith (1983) distinguish inductive inference (i.e., inductive learning) from learning from examples. They say that work in artificial intelligence (i.e., learning from examples) is more concerned with cognitive modeling than the work in inductive inference, and less concerned with formal properties such as convergence in the limit or computational efficiency. A learning algorithm is said to learn a concept in the limit if, after some finite number of examples, the learner's hypothesis is correct, and thereafter all the learner's hypotheses remain correct (Laird 1987). Convergence in the limit does capture the notion that the learner will eventually discard any false hypotheses, and that in finite time this progression will ultimately converge to a fixed, correct rule. The computational complexity of a learning algorithm is defined if and only if the algorithm converges (Angluin and Smith 1983). Computational efficiency can be considered as an additional important property of an inductive learning algorithm in practice. The construction of decision trees is an important type of inductive learning (Quinlan 1986; Mingers 1989a; Utgoff 1989). A decision tree (consisting of nodes and branches) represents a collection of rules, with each terminal

node corresponding to a specific decision rule. Decision trees are constructed beginning with the root of the tree and proceeding down to its leaves. This approach may be used directly for predictive or descriptive purposes (Braun and Chandler 1987; Carter and Catlett 1987; Messier and Hansen 1988; Shaw and Gentry 1990) or may be applied to knowledge acquisition for expert systems (Quinlan 1979; Michalski and Chilausky 1980; Quinlan 1987b). Decision tree induction is free from parametric and structural assumptions that most statistical methods, such as discriminant analysis, are based on. Researchers have found that large decision trees constructed from a training set usually do not retain their accuracy over the whole instance space (Quinlan 1983; Breiman et al. 1984; Spangler et al. 1989). Recently a number of papers have investigated pruning large decision trees built from training examples (Niblett 1987; Quinlan 1987a; Fisher and Schlimmer 1988; Mingers 1989b).

Many of the branches of the constructed decision trees will reflect chance occurrences in the particular data rather than representing true underlying relationships. Often, these are very unlikely in further examples. Since these less reliable branches can be removed by pruning, the pruned decision tree often gives better classification over the whole instance space even though it may have a higher error over the training set.

While these papers have reported excellent empirical results for pruning in terms of improving the accuracy of the learned concepts, those results depend heavily on the specific training set and the domains over which they apply. Whether these results hold in general and to what extent pruning improves a concept are unknown.

In this study we investigate various theoretical aspects of pruning a decision tree. We first review learning theory and previous research on decision tree induction and pruning. Then we determine the pruning error in a typical situation and determine the number of examples required to assure a desired confidence level. We also estimate the accuracy of a pruned decision tree for the learning situations where acquiring enough examples is difficult or even impossible. Finally we investigate conditions that give rise to increased accuracy due to pruning. (Each term will be rigorously defined later.)

CHAPTER 2 THEORETICAL BACKGROUND AND RELATED RESEARCH

2.1 Learning Theory

Since the early 1980s, the learning theories have been developed rapidly. Mitchell's version space (Mitchell 1982), Valiant's PAC (Probably Approximately Correct) learning (Valiant 1984; Angluin and Laird 1988), and Haussler's researches (Haussler 1988, 1990) are important building blocks for the learning theory. In general, learning theory considers three aspects of the learning process: concept accuracy, storage efficiency, and computational efficiency. In this section we summarize main stream learning theory in terms of these three important considerations.

2.1.1 Mitchell's Version Space

Mitchell (1982) gives an elegant framework for viewing the process of learning from examples and illustrates this framework by analyzing the process of learning simple conjunctive concepts. We start with basic definitions and terminology used in his framework.

Definition 2.0:

1. The instance space: The instance space is the space of all objects of interest. Each instance of a concept can be expressed in a feature-based or attribute-based form. Attribute-based instance spaces can be defined by the values of a fixed set of attributes, not all of which are necessarily relevant. Feature (or structure)-based instance spaces can be defined by allowing each instance to include several objects, each with its own attributes, and allowing binary relations that define a structure between objects.

For example, define an instance space consisting of the following attributes and binary relations (Haussler 1989). Permissible values appear in parentheses.

Attributes: . size (small, medium, large)
 . shape (convex, nonconvex)

Binary relations:

. distance-between (touching, nontouching)
. relative-position (on-top-of, under)

Then (size=small, shape=convex) is an example of an instance of an attribute-based instance space. The following instance is an example of a structure-based instance space.

(size=small, shape=convex)
(under, touching) ↑ ↓ (on-top-of, touching)
(size=large, shape=nonconvex)

2. Hypothesis space: The hypothesis space is the space of all plausible hypotheses. It is often called the rule space.

3. Inductive bias: A mechanism whereby the space of hypotheses is restricted or whereby some hypotheses are preferred, a priori, over others reflects the inductive bias.
4. Conjunctive concepts: Concepts described by logical expressions involving only conjunctions (i.e., AND operations) are called conjunctive concepts.
5. Disjunctive concepts: Concepts described by logical expressions involving only disjunction (i.e., Inclusive OR operations) are called disjunctive concepts.
6. Target concept: The target concept is the true concept. A learning system tries to find the target concept.

Mitchell's framework of learning can be described as follows. Let us assume that we are trying to learn some unknown target concept, f , defined on the instance space. This target concept can be any subset of the instance space. This may or may not be a conjunctive concept. Assume we have a set of examples of this target concept. Each example is generated by "sampling with replacement", and is one of following two cases.

Case 1. An instance satisfying the target concept.

Case 2. An instance not satisfying the target concept.

An example in Case 1 is called a "positive example", and an example in Case 2 is called a "negative example". We label each example accordingly. We call these labelled examples a sample, S , of the target concept.

We assume a hypothesis space, H , restricted to only conjunctive concepts. This is an inductive bias. Then the task is to produce a conjunctive concept that is consistent with the sample or to detect when no conjunctive concept is consistent with the sample. By "consistent" we require that a concept contain all instances of positive examples and no negative examples.

The version space is the set of all hypotheses $h \in H$ that are consistent with the sample. Since the version space depends on the hypothesis space, we denote the version space with respect to the hypothesis space H . The version space is empty in the case that no hypothesis in H is consistent with the sample.

Mitchell shows that the learning task (and related tasks) of producing a conjunctive concept consistent with the sample can be solved by keeping track of only two subsets of the version space--the set of the most specific hypotheses and the set of the most general hypotheses. These sets are updated accordingly as new examples are given.

Here we consider a finite instance space, and we assume no examples are in contradiction each other. There are two cases for the target concept f .

Case 1: The target concept $f \in H$

Case 2: The target concept $f \notin H$

For Case 1, the version space reduces until it contains only the target concept f , as examples are added to the

version space. For Case 2, the version space reduces until it becomes the empty set. Note that for both cases, the version space reduces to an informative terminal state which can tell the result of the learning task. If we stop before one of the terminal states is produced, then the learning task is incomplete and the current result is not as useful.

We say that the version space is exhausted with respect to H (we abbreviate this as "w.r.t. H ") if the version space is reduced to one of the above terminal states (i.e., is reduced to either the target concept f or an empty set).

Consider the situation that the version space contains only one hypothesis h , where h is not the target concept. If the target concept f is not an element of the hypothesis space H , then this situation may occur. For this situation, we assume that it is always possible to generate a new example which eliminates h from the version space. The version space will be empty and then be exhausted.

Mitchell's approach to inductive learning is to sample until the version space is exhausted. Stopping short of an exhausted version space leaves one with incomplete learning. However, the two subsets of version space bound the space where the target concept may exist.

2.1.2 Problems of Mitchell's Framework

There are two practical problems with Mitchell's approach. The first is that it may require too many examples to exhaust the version space (Haussler 1988).

The other problem is that even if we monitor only the two sets, the set of the most specific hypotheses and the set of the most general hypotheses, the storage needed can still become exponentially large as we build up examples (Haussler 1988).

These problems with the version space approach are overcome by incorporating probabilistic ideas (Valiant 1984). Here we will not require the complete exhaustion of the version space. Instead, we will require that a version space is "probably almost exhausted". (This term will be formally defined later.) This idea will do away with the first problem.

To handle the second problem, we will not try to remember the exact version space. Instead, we will require that any hypothesis from an "almost exhausted" version space will accurately approximate the target concept.

Hence, we replace Mitchell's idea of remembering all consistent hypotheses by a more elegant idea of drawing enough examples needed for a "probably almost exhaustion" of the version space and then finding an hypothesis (or hypotheses) consistent with these examples.

The idea of "almost exhausted" is formalized in Definition 2.1.

Definition 2.1: Given a hypothesis space H , a target concept f , a sequence of examples S of f , and an error tolerance ϵ , where $0 < \epsilon < 1$, the version space of S (w.r.t. H) is ϵ -exhausted (w.r.t. f) if it does not contain any hypothesis that has error ("error" will be carefully defined later) more than ϵ with respect to f .

2.1.3 Efficient PAC Learning

Now we introduce a formal definition of Probably Approximately Correct (PAC) learning based on the idea of an "almost exhausted" version space defined in the previous subsection. The concept of computational efficiency is also very important for a learning algorithm to be practical for larger problems. The PAC learning paradigm was introduced by Valiant in 1984. This model requires that a polynomial bounded algorithm identify a concept using a random sample, whose size is polynomially bounded, such that a learned concept has a high probability of being close to the true concept. Angluin and Laird (1988) coined the terminology PAC learning. A more precise definition follows.

Let X be the instance space of interest. The target concept f maps X into $\{0,1\}$. Similarly, for any other concept h , we have

$$h: X \rightarrow \{0,1\}.$$

The error, $d(h,f)$, of a learned concept h is the probability of the instances incorrectly classified by h . That is,

$$d(h,f) = \text{Prob}\{ x \in X: h(x) \neq f(x) \}.$$

$\text{Prob}\{\}$ is determined by an arbitrary sampling distribution, D , over X . Learning is accomplished by processing a learning procedure on a sample of instances called the training sample. Sampling is assumed to be with replacement with samples drawn independently.

For $0 < \epsilon, \delta < 1$, a learning procedure is said to be a probably approximately correct (with respect to D) identification of the target concept f if

$$\text{Prob}\{ d(h,f) \geq \epsilon \} \leq \delta.$$

We say that above learning procedure is an efficient PAC identifier if it is a polynomially bounded algorithm which identifies a concept from a random sample, whose size is polynomially bounded.

As we see in the definition, the PAC learning model is defined for $\{0,1\}$ -valued functions. Recently, Haussler (1990) gives a generalization of the PAC learning model that is based on statistical decision theory and can be applied for multi-valued discrete functions, real-valued functions and vector-valued functions. In this generalized model the learning system receives randomly drawn examples, each example consisting of an instance $x \in X$ and an outcome $y \in Y$. The learning system finds a hypothesis

$$h: X \rightarrow A$$

that specifies the appropriate action $a \in A$ to take for each instance x , in order to minimize the expectation of a loss function $L(y,a)$. Here X , Y and A are arbitrary sets, L is a real-valued function, and examples are generated according to arbitrary joint distribution on $X \times Y$.

2.1.4 The Performance of a Learning Algorithm

As we have seen in the definition of PAC learning, two measures of learning performance are relevant. The first is sample complexity, and the second is computational complexity.

1) Sample Complexity: The sample complexity is the number of random examples needed to produce a hypothesis that with high probability has small error. It is defined by taking the number of random examples needed in the worst case over all the target concepts in the class and all the probability distributions on the instance space.

2) Computational Complexity: The computational complexity is the worst case computation time to produce an hypothesis from a sample of a given size.

We use big-O notation to denote both complexities.

Vapnik (1982) was the first to give a characterization of the sample complexity of a learning algorithm. Below Lemma 2.2 gives a sufficient sample size for PAC identification. (Also see Blumer et al. 1987a.)

Lemma 2.2: Let N be the number of rules in the hypothesis space H . Let f be the target concept. If h is any hypothesis that agrees with at least

$$m = (1/\epsilon) \ln(N/\delta)$$

random samples, then

$$\text{Prob}\{ d(h, f) \geq \epsilon \} \leq \delta.$$

However, the above bound is very loose, and if the size of an hypothesis space is not finite, such as the set of intervals on the real line, the method cannot be applied.

So, there is a need to improve this bound. To improve the sample complexity we may use several different measures of a hypothesis space other than the number of rules in the hypothesis space (Vapnik 1982; Haussler 1988). Two other combinatorial parameters which measure the characteristics of a hypothesis space are given below.

Definition 2.3: (Growth function, VC dimension)

1. Growth function ($\pi_H(m)$): The growth function, $\pi_H(m)$, is the maximum number of dichotomies (i.e., the maximum number of ways of partitioning a set into a set of positive instances and a set of negative instances) induced by hypotheses in H on any set of m instances.
2. Vapnik-Chervonenkis dimension of H ($\text{VCdim}(H)$): Let I be a set of instances in X . If H induces all possible $2^{|I|}$ dichotomies of I , then we say that H shatters I . The Vapnik-

Chervonenkis dimension of H , denoted by $VCdim(H)$, is the cardinality of the largest finite subset I of X that is shattered by H , or equivalently, the largest m such that the Growth function $\pi_H(m) = 2^m$. If arbitrarily large subsets of X can be shattered, then $VCdim(H) = \infty$.

Below we give some examples of the above three measures for the hypothesis space, and give an illustration of Lemma 2.2.

Consider the following attributes of a firm. Suppose we are to characterize successful firms using the following list of attributes.

Attributes	Values
INDUSTRY TYPE	ELECTRONICS BANKING AUTOMOBILE
SIZE	LARGE MEDIUM SMALL
STRATEGIC PLANNING DEPARTMENT	YES NO
MIS DEPARTMENT	YES NO
CURRENT RATIO	GREATER THAN 3.0 BETWEEN 1.5 AND 3.0 LESS THAN 1.5
DEBT-EQUITY RATIO	GREATER THAN 0.7 LESS THAN OR EQUAL TO 0.7

Suppose H is a conjunctive concept. Since each attribute can either be a term of a conjunctive concept or not, the number of rules in H is $|H| = 3^3 4^3 = 1,728$. That is, $N = 1,728$.

Hence, by Lemma 2.2, the learned concept h has error ϵ with probability $1 - \delta$ after

$$\begin{aligned} & (1/\epsilon)(\ln(1/\delta) + \ln 1728) \\ & = (1/\epsilon)(\ln(1/\delta) + 7.455) \end{aligned}$$

random independent examples, regardless of the underlying distribution governing the generation of these examples. Note that the number of examples required grows slowly compared to the size of the hypothesis space.

Now suppose we wish to learn the range of liabilities to assets ratios within which successful companies operate. The instance space X is the interval $[0, 1]$. Let the hypothesis space H be the intervals $[x, y]$ with $0 \leq x \leq y \leq 1$ plus the empty set. Since there are an infinite number of intervals in $[0, 1]$, $|H| = \infty$. The growth function for H is determined as follows.

Consider the single example with value 0.6. The instance $0.6 \in X$ can be labelled as + (a positive example) by the concept $[0.5, 1]$ and - (a negative example) by the concept $[0, 0.5]$. Hence $\pi_H(1) = 2 = 2^1$.

Consider two examples with values 0.3 and 0.6. The following four concepts give all different partitionings of two examples.

$[0.1, 0.4]$ gives $(+,-)$;

$[0.4, 0.7]$ gives $(-,+)$;

$[0.2, 0.7]$ gives $(+,+)$; and

$[0.4, 0.5]$ gives $(-,-)$.

Thus $\pi_H(2) = 4 = 2^2$.

Now consider three examples with values a , b , and c , where $a < b < c$. Since no disjunction of intervals is allowed, no concept in H can classify (a, b, c) as $(+,-,+)$, but all other classification combinations are possible.

Thus $\pi_H(3) = 7 < 2^3$.

Therefore, $\text{VCdim}(H) = 2$.

We can improve the sample complexity of a learning algorithm when $\text{VCdim}(H)$ is considerably less than $\ln|H|$. The following lemma gives one of the main results using the VC dimension. (See Blumer et al. 1987b and Haussler 1988 for more details)

Lemma 2.4: (Blumer et al. 1987b) Let H be any nonempty hypothesis space, and let d be the VC dimension of H , where d is finite. Let f be the target concept. For any $0 < \epsilon < 1$, if h is any hypothesis that agrees with at least

$$m = \max \{ (4/\epsilon) \log(2/\delta), (8d/\epsilon) \log(13/\epsilon) \}$$

random samples, then

$$\text{Prob}\{ d(h, f) \geq \epsilon \} \leq \delta.$$

2.2 Decision Tree Induction Methods

A decision tree can be expressed as a disjunctive concept. Each path in a decision tree corresponds to a conjunction of variables (with values), and all pathes having the same class in their leaves can be combined with disjunctions. For example,

{ ((Outlook=sunny) and (Humidity=normal)) or (Outlook=overcast) or ((Outlook=rain) and (Windy=false)) }

is a disjunctive concept, P , represented by a decision tree in Figure 2.1 in Subsection 2.2.2.

2.2.1 Learning Disjunctive Concepts

We begin with the definition of the DNF (Disjunctive Normal Form) concept.

Definition 2.5: A disjunctive normal form (DNF) expression is any sum $m_1 + m_2 + \dots + m_r$ of monomials where each monomial m_i is a product of literals. Here "sum" means an inclusive OR operation and "product" means an AND operation. A literal is either a variable (i.e., an attribute with value) or the negation of a variable. An expression is monotone if no variable is negated in it.

For the case of monotone DNF expressions, having a bound, k , on the length of each disjunct (we call this a k -DNF

concept), Valiant (1984, 1985) showed that a PAC concept can be learned in polynomial time from negative examples with the sample complexity $O(n^k)$, where n is the number of variables. Haussler (1988) improves this result by using a dual greedy method which is a variant of the "star" methodology of Michalski (1983). The improved bound is $O((\log kn)^2)$.

Rivest (1987) showed k -DNF concepts without a monotonicity restriction are polynomially learnable using decision lists. A decision list is an extended "if-then-elseif-else" rule, where the tests in "if" parts are conjunctions of literals drawn from $2n$ literals. (See Rivest 1987 for the precise definition.) Compared to decision trees, decision lists have a simpler structure, but the complexity of the decisions allowed at a node is greater. Let k -DL be the set of all Boolean functions defined by decision lists, where each function in the list is a term of size at most k . k -DNF(n) is a proper subset of k -DL(n), where n is the number of variables used in the expression. Rivest shows that k -DL is polynomial-sized and polynomially-identifiable. If a class of formulae is polynomial-sized and polynomially identifiable, then it is polynomially learnable. However, computation of the "shortest" decision list consistent with a given sample is an NP-hard problem (Rivest 1987).

2.2.2 Learning a Decision Tree

A decision tree has much expressive power in the sense that it is concise and that there is no limitation on the attributes and classifications allowed, and is a more complex structure than the DNF concepts or decision lists. Therefore little theoretical research has been done on decision trees, and most research on learning a decision tree is based on heuristic reasoning. In this subsection we give a typical example of learning a decision tree and review previous work on decision tree induction.

Learning a decision tree requires a training set and a learning algorithm. The training set in Table 2.1 was given by Quinlan (1986). In Figure 2.1, a decision tree was produced from Table 2.1 by Quinlan's (1986) ID3 algorithm.

Here we have four attributes having values in parentheses.

1. Outlook (sunny, overcast, rain)
2. Temperature (hot, mild, cool)
3. Humidity (high, normal)
4. Windy (true, false)

Each line in Table 2.1 corresponds to an example.

The decision tree in Figure 2.1 can be interpreted as a set of five production rules.

Rule 1: If (Outlook = sunny) and (Humidity = high)

Then N

Table 2.1: Example training set

ATTRIBUTES				Member
Outlook	Temperature	Humidity	Windy	of
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

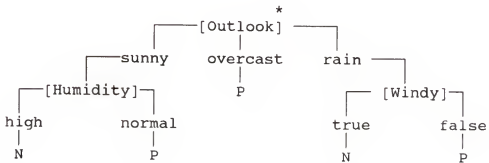


Figure 2.1: A decision tree learned from a training set in Table 2.1.

*: [] denotes an attribute.

Rule 2: If (Outlook = sunny) and (Humidity = normal)

Then P

Rule 3: If (Outlook = overcast)

Then P

Rule 4: If (Outlook = rain) and (Windy = true)

Then N

Rule 5: If (Outlook = rain) and (Windy = False)

Then P

There are a number of algorithms for learning decision trees (Quinlan 1979, 1983, 1986, 1987a; Niblett 1987; Utgoff 1989; Mingers 1989a, 1989b). Most algorithms involve three main stages:

- 1) Construct a complete tree able to exactly classify all the examples.

- 2) Prune this tree to give statistical reliability.

- 3) Process the pruned tree to improve understandability.

Some algorithms adopt pruning techniques while they construct a decision tree. Several construction-time pruning methods will be given in 2.2.2.3.

2.2.2.1 Decision tree construction

Decision tree construction can be performed incrementally or nonincrementally.

- 1) Nonincremental induction: A nonincremental algorithm infers a concept once, based on the entire set of available

training instances. Quinlan (1979, 1986) developed a non-incremental algorithm, ID3, for inducing a decision tree.

The overall approach to constructing a decision tree is to choose an attribute that best divides the examples into their classes, and then partition the data according to the values of that attribute. This process is recursively applied to each partitioned subset with the procedure terminating when all examples in the current subset have the same class. (This is a termination criterion. We may change this strict termination criterion depending on the situation.)

ID3 can build a decision tree by using the whole training set. However, since this approach is often computationally inefficient, ID3 often uses an iterative procedure. In this iterative framework, a subset of the training set, called the window, is chosen at random and a decision tree formed from it by using the overall approach of building a decision tree. This tree correctly classifies all examples in the window. All other examples in the training set are then classified using the tree. If the tree correctly classifies all other examples, then it is correct for the entire training set and the process terminates. If it does not correctly classify all other examples, a selection of the incorrectly classified examples is added to the window and the process repeated.

Quinlan's (1986) result showed that, in this way, correct decision trees have been found after only a few iterations for training sets of up to 30 thousand examples described in terms

of up to 50 attributes. Empirical evidence suggests that a correct decision tree is usually found more quickly by this iterative method than by forming a tree directly from the entire training set (Quinlan 1986; Wirth 1988). Note that the iterative framework cannot guarantee that "better" trees have not been overlooked. By better we mean the tree is "more general". That is, a better tree is a more comprehensive and concise description of decision rules for the situation.

The choice of an attribute for the root of the tree is crucial if the decision tree is to be simple. Since Quinlan's original work, there have been a number of alternative suggestions for measures to be used in selecting attributes. Some of them are as follows.

a) Quinlan's information measure: Quinlan(1979,1983) proposed an evaluation function based on a classic formula from information theory that measures the theoretical information content of a code. The measure is

$$- \sum p_i \log(p_i)$$

where p_i is the probability of the i -th message. The value of this measure depends on the likelihood of the various possible messages. If they are equally likely (and so the p_i are equal), there is the greatest amount of uncertainty and the information gained will be greatest. The less equal the probabilities, the less information there is to be gained. The value of the function also depends on the number of possible messages.

b) The chi-square contingency table statistic (Mingers 1986, 1987): This is the traditional statistic for measuring the association between two variables in a contingency table. It compares the observed frequencies with the frequencies that one would expect if there were no association between the variables.

c) The G statistic (Mingers 1989a): The G statistic is defined as follows.

$$G = 2N * IM,$$

where N is the number of examples, IM is Quinlan's information measure.

d) Probability measures (Mingers 1989a): Instead of using the value of chi-square or G statistics by itself, this method computes the probability of such a value on the chi-square distribution with the assumption that the attribute and the classes have no relationship.

e) The GINI index of diversity (Breiman et al. 1984): Let p_i be the probabilities of each class. GINI function measures the "impurity" of an attribute with respect to the classes as follows:

$$\text{The general GINI function} = \sum p_i^2.$$

The larger the GINI value, the more impure is the attribute.

f) Gain-ratio measure (Quinlan 1986): Quinlan's (1986) gain-ratio measure of attribute a, $GR(a)$, is

$$GR(a) = IM(a)/IV(a),$$

where $IV(a)$ is the information value of attribute a . As we see in the formula, this measure incorporate the idea that an attribute itself has some information value.

g) Marshall correction (Marshall 1986): This correction factor multiplies any calculated measure by the product of row totals giving the sum of probabilities of each partition. The reason for this is that it is preferable to obtain a partition which is balanced. Let G be any measure. Then Marshall's corrected measure $G^* = G \cdot A \cdot B$ if A and B are row totals.

There are also various methods other than ID3 for decision tree construction.

h) Hyperplane cut: Koehler and Majthay (1988) merged AI induction and the classical discriminant analysis method. They generate a hyperplane to partition the training set recursively. So, by choosing a combination of more than one attribute as a node at each step, they show that the constructed decision tree has better classification power than ID3.

i) Attribute-Value pair: Spangler et al. (1989) devised a metric which uses attribute-value pairs to select the best partition. They generate strictly binary trees by choosing the best attribute-value pair, rather than the best attribute, at any choice point (also see Cheng et al. 1988).

j) Use of background knowledge: Nunez (1991) presented a decision tree induction algorithm which executes several types of generalization and at the same time reduces the

classification cost by means of background knowledge. The background knowledge contains an ISA hierarchy and the measurement cost associated with each attribute. Buntine (1989) gives an experimental result on the effectiveness of Bayesian classifiers.

2) Incremental induction: An incremental induction algorithm revises the current concept, whenever necessary, in response to each newly observed training instance. This is appropriate for learning tasks in which there is a stream of training instances. Utgoff's (1989) ID5 is an example of this type of algorithm for decision tree induction. Van de Velde (1990) proposes an incremental algorithm for the induction of decision trees which are topologically minimal in the sense that an attribute's activity is localized as much as possible in the tree.

There are a number of other directions of research on the effectiveness of various decision tree construction methods. Cheng et al. (1988) identified two causes of over-specialization in ID3--the irrelevant value problem and the missing value problem. They developed a new algorithm that avoids these problems. Quinlan and Rivest (1989) used the minimum description length principle to find a decision tree which minimizes the total information required to specify the class of all training examples. This approach can be also considered as a method of pruning. Chan (1989) described a binary classification tree (BCT) which is a system that learns

from examples and represents learned concepts as a binary polythetic decision tree. Polythetic trees differ from monothetic decision trees in that a logical combination of multiple (vs. a single) attribute values may label each tree branch. Their empirical result showed that BCT is more accurate than ID3. Chan and Wong (1990) presented a probabilistic inductive learning system and showed better performance than ID3 in terms of computational efficiency and classification accuracy. Utgoff and Brodley (1990) presented PT2, an algorithm which is incremental and searches for a multivariate split at each node. Gelfand et al. (1991) propose an iterative growing and pruning algorithm for classification trees. This method divides the data sample into two subsets and iteratively grows a tree with one subset and prunes it with the other subset, successively interchanging the role of the two subsets. Chou (1991) present an iterative algorithm that finds a locally optimal partition, which minimizes an expected loss, for an arbitrary loss function.

2.2.2.2 PAC-learning a decision tree

Here we determine a sample complexity for PAC-learning a decision tree. Consider a hypothesis space, H , of n attributes. Let C_i be the number of permissible values for attribute i , where $1 \leq i \leq n$. Then the total number of distinct instances in the domain X is

$$d = \prod_{i=1}^n C_i.$$

Since each distinct instance can have a positive or negative label, the total number of concepts in the hypothesis space is

$$|H| = 2^d.$$

Note that any sample of distinct instances can be shattered by H . Since the maximum number of possible distinct instances is d , the Vapnik-Chervonenkis dimension of the hypothesis space is

$$\text{VCdim}(H) = d.$$

To determine the sample size for (ϵ, δ) -learning, we may use the following theorem from Tsai and Koehler (1991).

Theorem 2.6: (Sample size for ID3: Tsai and Koehler 1991)

1. For any given ϵ and δ , with $0 \leq \epsilon, \delta \leq 1$, if the sample size is at least

$$\lceil \ln(1/\delta) + d \ln 2 \rceil / \epsilon,$$

then ID3 will ϵ -exhaust H with probability at least $1-\delta$.

2. For

$$0 < \epsilon < 1/2$$

then ID3 must use a sample size of at least

$$\max\{ \lceil (1-\epsilon)/\epsilon \rceil \ln(1/\delta), \lceil d[1-2(\epsilon(1-\delta)+\delta)] \rceil \}.$$

For example, suppose the number of attributes is five, and each attribute have five possible values. Further suppose we require $\epsilon = .1$ and $\delta = .01$. Then by Theorem 2.6

$$\max(41.45, 2443.75) = 2444$$

samples would be necessary.

If there are any real-valued attributes, then $VCdim(H)$ will be infinity since the number of permissible values for that attribute is infinite. Many decision tree induction algorithms, however, often modify the algorithm to handle this situation. For example, they divide the range of a real-valued attribute by half, or divide the range into a finite meaningful interval based on the set of values of an attribute obtained in the training sample. With this modification, Theorem 2.6 can be applied with finite $VCdim(H)$. Theorem 2.6 can be used for most decision tree induction algorithms which select one attribute at a time.

2.2.2.3 Pruning a decision tree

When a decision tree algorithm is used with "uncertain" data rather than deterministic data, the pruning stage is important to remove branches with little statistical validity. By uncertain we mean "having error in representing the true concept". Uncertainty in the data may be due to noise in the measurements or to the presence of factors (i.e., attributes) which are hidden or cannot be measured. In the following we will use the word "noisy data" interchangeably with "uncertain

data" since many researchers use "noisy data" in this situation. Recently a number of papers investigated pruning large decision trees (Niblett 1987; Quinlan 1987a; Fisher and Schlimmer 1988; Mingers 1989b). Many of the branches of the constructed large decision trees will reflect chance occurrences in the particular data rather than representing true underlying relationships. Often, these are very unlikely in further examples. Since these less reliable branches can be removed by pruning, the pruned decision tree often gives better classification over the the whole instance space even though it may have a higher error over the training set. Several empirical methods have been proposed for pruning decision trees. We distinguish construction-time pruning from pruning after building a fully-grown decision tree (i.e., post-pruning).

Construction-time pruning methods are mainly used to decide when to stop expanding a decision tree. These methods replace the old termination criterion to stop expanding a tree when all examples in the current subset are in the same class by a new termination criteria, which is related to the selection measure used in constructing a decision tree. Two common approaches in construction-time pruning are the threshold method and chi-square test method.

1) Threshold method: Let $SM(a)$ be the value of the current selection measure at node a . The threshold method chooses a threshold t by some criteria. If $SM(a)$ is less than

t, then it stop expanding a decision tree at that node. (See Spangler et al. (1989) and Breiman et al. (1984) for the details)

2) Chi-square test method: In this method, we assume that at a certain node, the subtree created at that node will have the same class distribution as the current tree. With that assumption, we can calculate the chi-square statistic for each attribute. We stop expanding a tree if the chi-square statistic is not significant at a predetermined significance level. (Quinlan (1983) chose .01 as the significance level)

There are also some other variations such as the cross validation method used in the Assistant program (Niblett 1987).

As indicated by Niblett (1987), construction-time pruning methods have a weakness in that the criterion to stop expanding a tree is being made on local information alone. That is, we cannot ignore the possibility that decendent nodes of the node "a" may have better discriminating power. Breiman et al. (1984) have also reached the same conclusion that looking for the right stopping rule is the wrong way of looking at the problem, and pruning upward in the right way is more satisfactory. The following methods of pruning use global information, and are used to prune a fully-grown decision tree.

3) Error-complexity pruning: Breiman et al. (1984) have developed a two-stage method for pruning. The first stage

generates a series of trees pruned by different amounts. The second stage selects one of these by examining the number of classification errors each of them makes on an independent data set (a test or holdout data set). In pruning, the error-complexity method must take account of both the number of errors and the complexity(size) of the tree.

4) Critical value pruning: Mingers'(1987) method relies on estimating the importance or strength of a node from calculations done in the tree construction stage. It specifies a critical value and prunes those nodes which do not reach the critical value, unless a node further along the branch does reach it. The larger the critical value selected, the greater the degree of pruning and the smaller the resulting tree.

5) Minimum-error pruning: Niblett and Bratko (1986) have developed a method to find a single tree which should, theoretically, give the minimum error rate when classifying independent sets of data. It is based on the assumption of equally likely classes. If there are k classes, and n is the number of examples, and n_c examples are of class c , then the expected error, E , is

$$(n - n_c + k - 1) / (n + k).$$

For this method, the number of classes strongly affects the degree of pruning, leading to unstable results (Mingers 1989).

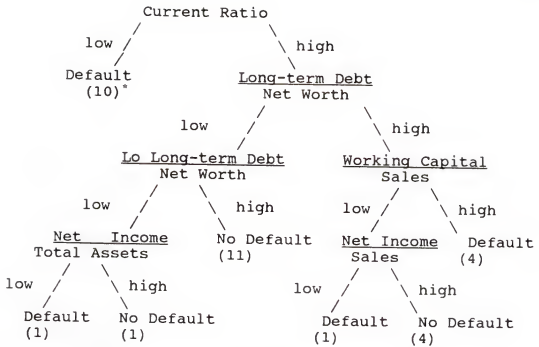
6) Reduced error Pruning: Quinlan (1987a) suggested a method which produces a series of pruned trees by using test data directly, rather than using it only for the selection of the best tree. This approach generates a set of trees, ending with the smallest minimum-error tree on the test data.

7) Pessimistic error pruning: Quinlan (1987a) suggested using the continuity correction for the binomial distribution to obtain a more realistic estimate of the misclassification rate.

In the following we give an example of pruning a decision tree. Messier and Hansen (1988) developed a decision tree using ID3 to provide rules to predict loan default by companies. They started with 18 financial ratios and trends for each of their training instances.

In Figure 2.2 we have altered their final tree to express all the variables as binary variables. The exact meaning of high and low for each attribute is given below:

Attributes	low	high
Current Ratio	< 1.912	≥ 1.912
Long-term Debt/Net Worth	$< .486$	$\geq .486$
Lo Long-term Debt/Net Worth	$< .046$	$\geq .046$ and $< .486$
Working Capital/Sales	$< .222$	$\geq .222$
Net Income/Total Assets	$< .100$	$\geq .100$
Net Income/Sales	$< .010$	$\geq .010$



*: The number of examples in each terminal node.

Figure 2.2: Decision tree predicting loan default

To give an example of pruning, we use the Error-complexity method of Breiman et al. (1984) and Minimum-error pruning of Niblett and Bratko (1986).

Define the error-complexity measure, α , for a non-terminal (nonleaf) node t as follows:

$$\alpha = (e_p - e_u) / (n_t - 1),$$

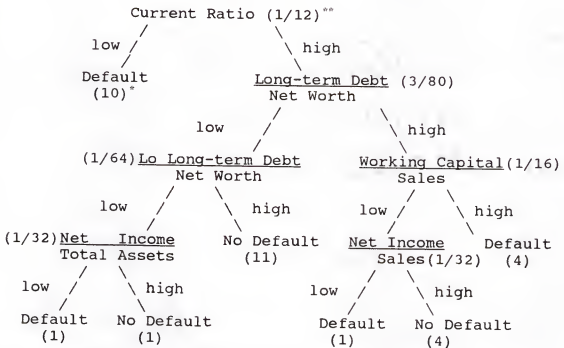
where e_p = the error rate when we prune node t ,

e_u = the error rate of the unpruned tree, and

n_t = the number of leaf nodes which are descendants of t .

The Error-complexity method calculates α for each nonterminal node and selects a node with the smallest value of α for pruning. (That is, find the weakest link in the tree). This

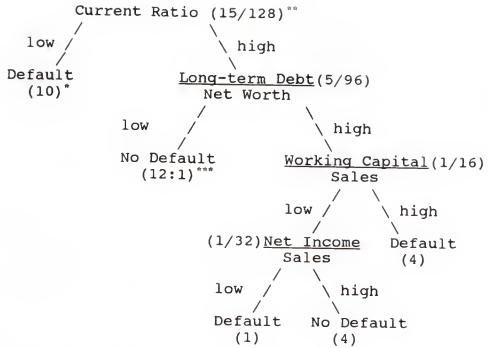
process is repeated and a tree is selected as the final tree based on the misclassification rate over an independent test set. In Figure 2.3 we show the α values for each node. Figure 2.4 shows the tree after pruning the node labelled Lo Long-term Debt/Net Worth, which had the smallest value of α .



*: The number of examples in each terminal node.

**: α values for each node.

Figure 2.3: Decision Tree with α values for each node



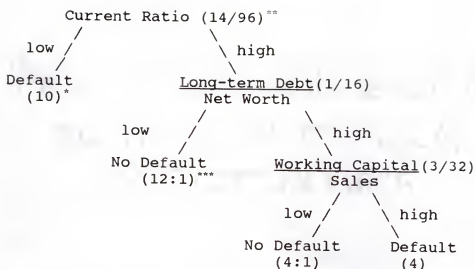
*: The number of examples in each terminal node.

** : α values for each node.

***: 12 examples are in No Default, One example is in Default.

Figure 2.4: Decision Tree with α values for each node

In Figure 2.4, the node, Net Income/Sales, has the smallest α value, and can be pruned. Figure 2.5 shows the tree after pruning the node labelled Net Income/Sales.



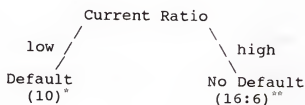
*: The number of examples in each terminal node.

**: α values for each node.

***: 12 examples are in No Default, One example is in Default.

Figure 2.5: A Pruned Decision Tree predicting Loan Default

Figure 2.6 shows the tree after pruning the node labelled Long-term Debt/Net Worth, which had the smallest value of α .



*: 10 examples are in Default.

**: 16 examples are in No Default, 6 examples are in Default.

Figure 2.6: A Pruned Decision Tree predicting Loan Default

Each of pruned trees is used to classify an independent test set, and the smallest tree with a misclassification rate within one standard error of the minimum is selected as the final tree (Breiman et al. 1984; Mingers 1989b).

Using the same hold-out sample of Messier and Hansen (1988), we have the following misclassification rates:

# of nodes in the pruned tree	misclassification rates
6 (Unpruned tree: Figure 2.2)	2/16
4 (Figure 2.4)	2/16
3 (Figure 2.5)	2/16
1 (Figure 2.6)	2/16
0	0 or 1 *

*: The misclassification rate depends on the labelling rule since there are equal number of Default and No Default examples.

Here we choose the tree with only a root node (Figure 2.6) as the best pruned tree. The above hold-out sample may not be considered as a random sample since these are all in one class, the Default class. In general, misclassification rates show the near-convex pattern with a random hold-out sample. (That is, the misclassification rate decreases as we prune a small amount, but increases as we prune too much).

Now consider the Minimum-error method of pruning. The expected error rate of the unpruned node is calculated by weighting the error rates of each branch according to the proportion of examples in each branch. For example, if we prune the node Net Income/Sales in Figure 2.2, then with the number of classes $k = 2$,

$$E = (5-4+1)/(5+2) = 2/7.$$

If we do not prune this node, the expected error rate is

$$E = (1/5)(1-1+1)/(1+2) + (4/5)(4-4+1)/(4+2) = 1/5.$$

Since pruning increases the expected error rate we do not prune the node Net Income/Sales. By the similar calculations for all nonterminal nodes we see that the unpruned tree in Figure 2.2 has the smallest expected error rate.

Mingers (1989b) presented empirical comparisons of the above five post-pruning methods across several domains. He used four selection measures--the G-statistic, the G-statistic with Marshall's correction, the probability of G from the Chi-square distribution, and the Gain-ratio measure--to investigate the interaction between the construction and pruning methods. The achievable accuracy differs markedly between domains, depending on their inherent uncertainty. He found that, for most domains, pruning improved the accuracy by 20% to 25%. The result shows that three methods--error-complexity, critical value and reduced error--perform well, while the other two may cause problems in terms of prediction accuracy. Minimum-error pruning produces the largest trees, and error-complexity and critical value produce the smallest decision trees. He also shows that there is no significant interaction between the construction and pruning methods. Quinlan (1987a) assessed the performance of pruning methods--error-complexity, reduced error, and pessimistic pruning--in terms of the clarity and accuracy of the pruned tree. He used

six domains to test including both real world tasks and synthetic tasks. The result shows that error-complexity pruning tends to produce smaller decision trees than either reduced error or pessimistic pruning. While all the pruned tree had superior or equivalent accuracy compared to the unpruned tree, the reduced error method showed slight superiority. (For other empirical results, see Niblett and Bratko 1986, and Clark and Niblett 1987).

Fisher and Schlimmer (1988) suggest that the benefits of pruning vary with the amount of training and with the statistical dependence of the concept members on the defining attributes.

2.3 Focus of Research

While these pruning methods have reported excellent empirical results in terms of improving the accuracy of the learned concepts, those results depended heavily on the specific training set and the domains over which they applied. Whether these results hold in general and to what extent pruning improves a concept are unknown.

In this study we focus on various theoretical aspects of pruning. In Chapter 3 we propose a particular type of pruning and determine its theoretical effect. This is combined with recent results in learning theory to produce a pruning method that will yield a concept that is probably approximately

correct (PAC). In Chapter 4 we focus on the accuracy of a concept learned with pruning. In Chapter 5 we present conditions under which pruning is useful and investigate the reason why pruning problems arise. Finally, in Chapter 6 we summarize our results and suggest areas for future research.

CHAPTER 3 PAC LEARNING A DECISION TREE WITH PRUNING

3.1 Introduction

In this chapter we focus on a particular type of pruning and determine its theoretical effect. This is combined with recent results in learning theory to produce a pruning method that will yield a concept that is probably approximately correct (PAC).

In Section 3.2 we present notation and give background material for constructing a decision tree with minimum rank. A pruning algorithm is motivated and presented in Section 3.3. In Section 3.4 we determine a bound on the error due to pruning. In Section 3.5 we give conditions for PAC identification with pruning. In Section 3.6 we discuss additional pruning rules. Finally, in Section 3.7 we summarize our results and suggest areas for future research.

Many algorithms are known for determining decision trees (Quinlan 1986; Cheng et al. 1988; Mingers 1989a; Utgoff 1989). Recently, Ehrenfeucht and Haussler (1988) presented the first PAC learning algorithm for binary decision trees of rank at most r (rank is defined below). For any fixed rank r , the number of random examples and computation time required in

this algorithm is polynomial in the number of attributes and linear in $1/\epsilon$ and $\log(1/\delta)$.

We take the rank as a conciseness measure of a decision tree and give a pruning algorithm which gives the least upperbound of a pruning error. We also determine the error introduced by pruning and give the required sample size to guarantee PAC-identification with accuracy parameter ϵ and confidence parameter δ .

3.2 Binary Decision Trees

3.2.1 Definitions

We begin with formal definitions of binary decision trees, their rank, the functions they represent and their error in an instance space. Our notation is similar to that given by Ehrenfeucht and Haussler (1988).

Definition 3.0 (A reduced binary decision tree): Let $V_n = \{v_1, \dots, v_n\}$ be a set of n Boolean variables. Let $X_n = \{0,1\}^n$. A binary decision tree is defined as follows:

(i) If Q is a tree with only a root labelled either 0 or 1, then Q is a binary decision tree over V_n (Below we abbreviate this case by saying " $Q=0$ " or " $Q=1$ ").

(ii) Let Q_0 be the left subtree of Q , and let Q_1 be the right subtree of Q . If the root node v of Q is in V_n and the left subtree Q_0 (a 0-Subtree), and right subtree Q_1 (a 1-

Subtree) are binary decision trees, then Q is also a binary decision tree.

We define an internal node i of a decision tree Q as a node in Q which has left and right subtrees. All nodes which are not internal nodes are called leaf nodes or simply leaves of a decision tree Q . We say that an internal node i is informative if it has only leaves and the leaves have different labels.

We say that a decision tree is reduced if each variable appears at most once in any path from the root to a leaf.

The level of node i is the number of predecessor nodes from the root. The height of a decision tree Q is defined as the maximum of the levels of all leaf nodes of Q .

A fully labeled tree is a tree for which every leaf has a 0 or 1 label.

A complete binary tree is a binary decision tree where every leaf is at the same level.

Definition 3.1 (A function and rank of a decision tree): A fully labeled binary decision tree Q represents a Boolean function f_Q defined as follows:

(i) If $Q = 0$ then f_Q is the constant function 0 and if $Q = 1$ then f_Q is the constant function 1.

(ii) Else if v_i is the label of the root of Q , then for any point $x = (a_1, \dots, a_n) \in X_n$, if $a_i = 0$ then $f_Q(x) = f_{Q_0}(x)$, else $f_Q(x) = f_{Q_1}(x)$.

The rank of a decision tree Q , denoted $r(Q)$, is defined as follows:

(i) If $Q=0$ or $Q=1$ then $r(Q)=0$.

(ii) Else if r_0 is the rank of the 0-subtree of Q and r_1 is the rank of the 1-subtree, then

$$r(Q) = \begin{cases} \max(r_0, r_1) & \text{if } r_0 \neq r_1 \\ r_0 + 1 \text{ (} = r_1 + 1 \text{)} & \text{otherwise} \end{cases}$$

Let T_n^r be the set of all binary decision trees over V_n of rank at most r and let F_n^r be the set of Boolean functions on X_n that are represented by trees in T_n^r .

The error of a decision tree Q is defined below.

Definition 3.2: Let f be the target concept. The error of a decision tree Q , denoted $e(Q)$ or e when Q is understood, is defined as the probability of all x such that $f(x) \neq f_Q(x)$. That is,

$$e(Q) = \text{Prob} \{ x : f(x) \neq f_Q(x) \}.$$

3.2.2 Finding Consistent Binary Decision Trees with Minimum Rank

There are several algorithms for finding a decision tree using a training set S (see Quinlan 1986, Cheng et al. 1988, Mingers 1989a, Spangler et al. 1989 and Utgoff 1989). Here we focus on a PAC-learning method given by Ehrenfeucht and Haussler (1988).

Definition 3.3: An example of a Boolean function f on X_n is a pair $(x, f(x))$, where $x \in X_n$. The example is positive if $f(x)=1$, else it is negative. A sample, S , of f is a set of examples of f . $|S|$ denotes the number of examples in S . A decision tree Q over V_n is consistent with a sample S if for any example $(x, f(x))$ in S , $f(x)=f_Q(x)$. The rank of a sample S , denoted $r(S)$, is the minimum rank of any decision tree that is consistent with S . We say a variable v in V_n is informative (on S) if sample S contains at least one example in which the label of v is 0 and at least one example in which the label of v is 1. Let S_0^v be the set of all examples $(x, f(x))$ such that the label of v is 0, and let S_1^v be the set of all examples $(x, f(x))$ such that the label of v is 1.

The following two algorithms can be used to determine a decision tree consistent with a sample S .

Definition 3.4: The procedure $\text{Find}(S, k)$ is reproduced from Ehrenfeucht and Haussler (1988).

Procedure $\text{Find}(S, k)$

Input: A nonempty sample S of some Boolean function on X_n and an integer k , $n \geq k \geq 0$.

Output: A binary decision tree of rank at most k that is consistent with S if one exists, else "none".

1. If all examples in S are positive, stop and return the decision tree $Q = 1$; if all examples are negative, stop and return $Q = 0$.
2. If $k = 0$, stop and return "none".
3. For each informative variable v in V_n
 - a. Let $Q_0^v = \text{Find}(S_0^v, k-1)$ and $Q_1^v = \text{Find}(S_1^v, k-1)$.
 - b. If both recursive calls are successful (i.e., neither $Q_0^v = \text{none}$, nor $Q_1^v = \text{none}$) then stop and return the decision tree with root labeled v , 0-subtree Q_0^v and 1-subtree Q_1^v .
 - c. If one recursive call is successful but the other is not, then
 - i) Re-execute the unsuccessful recursive call with rank bound k instead of $k-1$, (i.e., if Q_1^v is a tree but $Q_0^v = \text{none}$ then let $Q_0^v = \text{Find}(S_0^v, k)$).
 - ii) If the re-executed call is now successful, then let Q be the decision tree with root labeled v , 0-subtree Q_0^v and 1-subtree Q_1^v , else let $Q = \text{"none"}$.
 - iii) Stop and return Q .
4. Stop and return "none".

Definition 3.5: The procedure Findmin(S) is reproduced from Ehrenfeucht and Haussler (1988).

Procedure Findmin(S)

Input: A nonempty sample S of some Boolean function on X_n .

Output: A minimal rank reduced binary decision tree of S .

1. Repeat Find(S, k) for $k = 0, 1, 2, \dots$ until a decision tree is returned.
2. Stop and return Q .

Findmin()'s performance is given below.

Theorem 3.6 (A Decision tree with minimum rank: Ehrenfeucht and Haussler, 1988): Given a sample S of a Boolean function on X_n , using Findmin(S) we can produce a decision tree that is consistent with S and has rank $r(S)$ in time $O(|S|(n+1)^{2r(S)})$.

We will use Findmin() in our approach.

3.3 Pruning a Consistent Decision Tree to a Desired Rank

In this section we present a pruning algorithm that will be used with Findmin() to give certain theoretical results. We begin with definitions of pruning and labelling.

3.3.1 Definitions

There are many ways to prune a decision tree. We first address what we mean by pruning.

Definition 3.7: Pruning a decision tree Q at node v is defined as removing the left and right subtrees of node v , and

labelling the node (now a leaf) according to some labelling rule.

There are many possible labelling rules. For most of this chapter we focus on deterministic methods for labelling. Later, we will look at two nondeterministic rules.

Definition 3.8 (Deterministic Labelling): Let i be an internal node in a decision tree Q , and $Q(i)$ be a subtree of Q such that node i is the root of the tree $Q(i)$.

1. Sample labelling:

Let $s(i)$ be a subset of the sample S used to construct Q from its root to node i .

Let $s_0(i)$ denote the number of negative examples in $s(i)$ and $s_1(i)$ denote the number of positive examples in $s(i)$. If we prune Q at i , then

- (i) If $s_0(i) \geq s_1(i)$, label i as 0.
- (ii) If $s_0(i) < s_1(i)$, label i as 1.

2. Tree labelling:

Let v_{i_1}, \dots, v_{i_p} be the nodes in the path from the root to the parent of node i in Q . And let a_{i_1}, \dots, a_{i_p} be the labels of each node, where i_1, \dots, i_p are p distinct indices of $\{1, \dots, n\}$.

Let $l_0(i) = |\{x: f_Q(x)=0\}|$, where $x \in X_n$ such that a_{i_1}, \dots, a_{i_p} are the same.

Let $l_1(i) = |\{x: f_Q(x)=1\}|$, where $x \in X_n$ such that a_{i_1}, \dots, a_{i_p} are the same. If we prune Q at i , then

- (i) If $l_0(i) \geq l_1(i)$, label i as 0.
- (ii) If $l_0(i) < l_1(i)$, label i as 1.

Below we give an illustrative example where the above two methods give a different labelling.

Consider the case of an instance space over two Boolean variables. Further suppose we have five training examples. In Table 3.1 below, $n()$ denotes the number of examples for each point x . $f_Q()$ denotes the boolean function of the concept learned by constructing a consistent decision tree from these training examples.

Table 3.1: A Decision Tree with One Node

x	$n(x)$	$f_Q(x)$
(1,1)	1	1
(1,0)	0	1
(0,1)	1	1
(0,0)	3	0

Suppose we prune Q at the root. If we label the root node by the sample labelling rule, the label of the root is 0 since $s_0()=3 \geq s_1()=2$.

However, if we use the tree labelling rule, the label of the root will be 1 since $l_0()=1 < l_1()=3$.

3.3.2 Pruning

We now consider the following problem: Given a consistent decision tree with rank k , produce a pruned decision tree with rank r ($k > r$). We want a pruned tree to have the least amount of error due to pruning. We present an algorithm that solves this problem in time $O(2^{k+r} + |S|)$ using sample labelling and $O(2^{k+r} + (en/k)^k)$ using tree labelling.

For any decision tree Q of rank $k > r$, it is easily seen that one of the following mutually exclusive cases must hold.

Case 1: $r(Q_0) = k$ and $r(Q_1) < r$

Case 2: $r(Q_0) = k$ and $r \leq r(Q_1) < k$

Case 3: $r(Q_0) < r$ and $r(Q_1) = k$

Case 4: $r \leq r(Q_0) < k$ and $r(Q_1) = k$

Case 5: $r(Q_0) = r(Q_1) = k-1$

For a given decision tree of rank k , there may be many ways to prune it to rank r . Since our objective is to minimize the pruning error, one possible strategy is to prune the least number of nodes of a consistent decision tree Q . In other words, we will prune Q to the largest subtree having the desired rank. The idea of our pruning algorithm is as follows. In Cases 1 and 3, pruning only one subtree with rank k to rank r is enough to form a pruned decision tree with rank r . Pruning the other subtree with rank less than r to a lower rank is not needed since the resulting tree will still have same rank. Similarly, using this minimum node pruning rule we

have two possible alternative ways of pruning in Cases 2 and 4.

(i) Prune one subtree with rank k to rank $r-1$ and prune the other subtree to rank r .

(ii) Prune one subtree with rank k to rank r and prune the other subtree to rank $r-1$.

In the algorithm shown below, we use method (i) and in Section 3.4, we show that this method gives the least pruning error.

Finally, in Case 5, we will arbitrarily prune the tree by giving the preference to the 0-subtree.

3.3.3 A Pruning Algorithm

Below is pruning algorithm Prune(). This algorithm can prune any binary tree. However, we restrict our initial input to trees produced by Findmin(S) in order to later guarantee PAC identification. In the following $Q \leftarrow P$ means Q is replaced by P .

Procedure Prune(r, k, Q, S)

Input: A decision tree Q with rank at most k , a sample S
and an integer r such that $0 \leq r \leq k$.

Output : A decision tree with rank at most r .

Let $s_0(Q, S)$ = the number of examples in S such that $f_Q(x)=0$.

Let $s_1(Q, S)$ = the number of examples in S such that $f_Q(x)=1$.

Let $l_0(Q)$ = the number of instances $x \in X_n$ such that $f_Q(x)=0$.

Let $l_1(Q)$ = the number of instances $x \in X_n$ such that $f_Q(x)=1$.

1. If $r(Q) \leq r$, then stop and return Q .
2. If $r=0$, then stop and label Q by the appropriate labelling rule:

- 1) Sample labelling:

$Q=0$ if $s_0(Q,S) \geq s_1(Q,S)$, otherwise $Q=1$.

- 2) Tree labelling:

$Q=0$ if $l_0(Q) \geq l_1(Q)$, otherwise $Q=1$.

Return Q .

3. Let $k_0 = r(Q_0)$, $k_1 = r(Q_1)$.

Let S_0 be the set of all examples $(x, f(x))$ in S such that

$x=(a_1, \dots, a_n)$ and $a_i=0$, where $v_i \in V_n$ is the root of Q .

Let S_1 be the set of all examples $(x, f(x))$ in S such that

$x=(a_1, \dots, a_n)$ and $a_i=1$, where $v_i \in V_n$ is the root of Q .

Case 1: If $k_0 = k$ and $k_1 < r$

then $Q_0 \leftarrow \text{Prune}(r, k_0, Q_0, S_0)$.

Case 2: If $k_0 = k$ and $r \leq k_1 < k$

then $Q_1 \leftarrow \text{Prune}(r, k_1, Q_1, S_1)$, $Q_0 \leftarrow \text{Prune}(r-1, k_0, Q_0, S_0)$

Case 3: If $k_0 < r$ and $k_1 = k$

then $Q_1 \leftarrow \text{Prune}(r, k_1, Q_1, S_1)$.

Case 4: If $r \leq k_0 < k$ and $k_1 = k$

then $Q_0 \leftarrow \text{Prune}(r, k_0, Q_0, S_0)$, $Q_1 \leftarrow \text{Prune}(r-1, k_1, Q_1, S_1)$

Case 5: If $k_0 = k_1 = k-1$

then $Q_0 \leftarrow \text{Prune}(r, k_0, Q_0, S_0)$, $Q_1 \leftarrow \text{Prune}(r-1, k_1, Q_1, S_1)$

4. Stop and return Q .

To obtain our theoretical results it will be useful to have another pruning algorithm. This alternative pruning method prunes one subtree of rank k to rank r and prunes the other subtree to rank $r-1$. We define $W\text{-Prune}()$, as this alternative pruning algorithm.

Procedure $W\text{-Prune}(r, k, Q, S)$

Input: A decision tree Q with rank at most k , a sample S
and an integer r such that $0 \leq r \leq k$.

Output : A decision tree with rank at most r .

Let $s_0(Q, S)$ = the number of examples in S such that $f_Q(x)=0$.

Let $s_1(Q, S)$ = the number of examples in S such that $f_Q(x)=1$.

Let $l_0(Q)$ = the number of instances $x \in X_n$ such that $f_Q(x)=0$.

Let $l_1(Q)$ = the number of instances $x \in X_n$ such that $f_Q(x)=1$.

1. If $r(Q) \leq r$, then stop and return Q .
2. If $r=0$, then stop and label Q by the appropriate labelling rule:

- 1) Sample labelling:

$Q=0$ if $s_0(Q, S) \geq s_1(Q, S)$, otherwise $Q=1$.

- 2) Tree labelling:

$Q=0$ if $l_0(Q) \geq l_1(Q)$, otherwise $Q=1$.

Return Q .

3. Let $k_0 = r(Q_0)$, $k_1 = r(Q_1)$.

Let S_0 be the set of all examples $(x, f(x))$ in S such that $x=(a_1, \dots, a_n)$ and $a_i=0$, where $v_i \in V_n$ is the root of Q .

Let S_1 be the set of all examples $(x, f(x))$ in S such that

$x = (a_1, \dots, a_n)$ and $a_i = 1$, where $v_i \in V_n$ is the root of Q .

Case 1: If $k_0 = k$ and $k_1 < r$

then $Q_0 \leftarrow W\text{-Prune}(r, k_0, Q_0, S_0)$.

Case 2: If $k_0 = k$ and $r \leq k_1 < k$

then $Q_1 \leftarrow W\text{-Prune}(r-1, k_1, Q_1, S_1)$, $Q_0 \leftarrow W\text{-Prune}(r, k_0, Q_0, S_0)$

Case 3: If $k_0 < r$ and $k_1 = k$

then $Q_1 \leftarrow W\text{-Prune}(r, k_1, Q_1, S_1)$.

Case 4: If $r \leq k_0 < k$ and $k_1 = k$

then $Q_0 \leftarrow W\text{-Prune}(r-1, k_0, Q_0, S_0)$, $Q_1 \leftarrow W\text{-Prune}(r, k_1, Q_1, S_1)$

Case 5: If $k_0 = k_1 = k-1$

then $Q_0 \leftarrow W\text{-Prune}(r, k_0, Q_0, S_0)$, $Q_1 \leftarrow W\text{-Prune}(r-1, k_1, Q_1, S_1)$

4. Stop and return Q .

Lemma 3.9 shows that $\text{Prune}(r, k, Q, S)$ and $W\text{-Prune}(r, k, Q, S)$ always give a pruned tree with the desired rank at most r .

Lemma 3.9: The procedures $\text{Prune}(r, k, Q, S)$ and $W\text{-Prune}(r, k, Q, S)$ are correct.

Proof: The proof is by induction on r . If $r=0$ and $k \geq 0$ then $\text{Prune}()$ and $W\text{-Prune}()$ return $Q=0$ or $Q=1$. Hence, $r(Q) = 0 = r$.

Now suppose $\text{Prune}()$ and $W\text{-Prune}()$ run correctly on all cases requiring rank $r' \leq r-1$ and $k \geq 0$. In each of the five cases treated within $\text{Prune}()$ and $W\text{-Prune}()$, either a tree of rank less than or equal to $r-1$ is to be produced, or a tree of

rank at most r is to be produced. By assumption, for those requiring rank less than or equal to $r-1$, the algorithm will work correctly. So we turn our attention to the case requiring rank at most r .

Without loss of generality, suppose Q_0 requires rank at most r . $\text{Prune}(r, k_0, Q_0, S_0)$ and $\text{W-Prune}(r, k_0, Q_0, S_0)$ will operate on a tree with one fewer variable than Q . Since the tree is reduced, the number of variables used in Q_0 from its root to leaves must be less than or equal to $n-1$. Since only one subprocedure requires a tree with rank at most r in each recursion, we divide that subprocedure into subprocedures until the subprocedure requiring rank at most r will be operated on a tree with rank less than or equal to r . So, ultimately in the recursion, all subprocedures except one require trees with rank less than or equal to $r-1$, and the subprocedure requiring a tree with rank at most r will be operated on a tree with rank less than or equal to r since the number of variables used in the tree is reduced enough to guarantee that the rank of the tree is less than or equal to r . If $r(Q) \leq r$, by Step 1, the $\text{Prune}()$ and $\text{W-Prune}()$ return a tree with rank at most r correctly. Since $\text{Prune}()$ and $\text{W-Prune}()$ are correct for all other subprocedures requiring rank less than or equal to $r-1$, by induction, the procedure $\text{Prune}(r, k, Q, S)$ and $\text{W-Prune}(r, k, Q, S)$ return a pruned tree with rank at most r .

Therefore, the procedure $\text{Prune}(r, k, Q, S)$ and $W\text{-Prune}(r, k, Q, S)$ are correct for all $0 \leq r \leq k$.

□

Lemma 3.10 will be used to prove the time complexity of the procedure $\text{Prune}()$.

Lemma 3.10 (An upperbound on the number of nodes in a reduced decision tree over V_n : Ehrenfeucht and Haussler, 1988): Let j be the number of nodes in a reduced decision tree over V_n of rank k , where $n \geq k \geq 1$. Then

$$j < 2(en/k)^k,$$

where e is the base of the natural logarithm.

Lemma 3.11: For any nonempty tree Q with rank $k \geq 0$, the time of $\text{Prune}(r, k, Q, S)$ is $O(2^{k+r} + |S|)$ for sample labelling, and $O(2^{k+r} + (en/k)^k)$ for tree labelling.

Proof: $\text{Prune}()$ will stop either when $r=0$ or when k reduces to r . For each call of $\text{Prune}()$, r or k will be reduced by at least one and at most two new recursive calls of $\text{Prune}()$ are made. Since we do not have subprocedures if $r+k \leq 2$, there are at most $r+k-2$ steps. So, the total number of calls of $\text{Prune}()$ will be at most 2^{k+r-1} . For each call of $\text{Prune}()$, there are a constant number of unit operations (mainly, comparisons) except for the labelling time. The time for the labelling

step (Step 2) depends on the labelling rule. If we label a pruned node using sample labelling, at most $|S|$ additional time is needed since S_0 and S_1 in the Prune() are disjoint subsets of S and labelling occurs at most once in each subprocedure. If we label a pruned node using tree labelling, at most $(en/k)^k$ additional time is needed since Q_0 and Q_1 are disjoint subtrees of Q and labelling occurs at most once in each subprocedure. Also, the number of leaf nodes which we need to scan to label the node is less than or equal to $(en/k)^k$ by Lemma 3.10. So, the time of Prune(r, k, Q, S) is $O(2^{k+r} + |S|)$ for sample labelling, and $O(2^{k+r} + (en/k)^k)$ for tree labelling.

□

Lemma 3.11 gives the time complexity of this algorithm. If we use the sample labelling rule, the time required for pruning is independent of the number of variables, n , and far less than the tree construction time of Findmin() in Theorem 3.6 since $n \geq 1$, $k \geq r$ and $|S| \geq 1$. If we use the tree labelling rule, the time required for pruning is polynomial in n for fixed k and less than the time of Findmin() since $|S| \geq 1$.

3.4 Determining the Pruning Error

To determine a PAC learning result for pruning, we need to bound the pruning error. Here we give a formal definition of the pruning error and the least upperbound of the pruning error for a given rank tree. We can define the pruning error over the training sample or over the whole instance space. Here we use the whole instance space. The pruning error over the training sample will always be positive since we are assuming that the starting decision tree is reduced and is consistent with the training sample. However, if we define the pruning error over the instance space, we may get different results since the pruned tree may perform better than the unpruned tree as shown in many studies (Quinlan 1987a; Mingers 1989b).

Definition 3.12: Given a decision tree $P(Q)$ formed by pruning a decision tree Q produced by Findmin(), the pruning error $e(Q, P(Q))$, is defined as the difference between the error of $P(Q)$ and the error of Q .

$$\begin{aligned} e(Q, P(Q)) &= e(P(Q)) - e(Q) \\ &= \text{Prob} \{ x : f(x) \neq f_{P(Q)}(x) \} - \text{Prob} \{ x : f(x) \neq f_Q(x) \}. \end{aligned}$$

For an arbitrary probability distribution D and any decision tree Q in which all nodes are informative, we can easily verify that the pruning error is in the range $(-1, 1)$

since, by definition, the pruning error is the difference of two probabilities, where $e(Q)$ is in the range $[0,1)$ and $e(P(Q))$ is in the range $(0,1)$. Since we assume that all examples are drawn according to D without error and $\text{Findmin}(S)$ guarantees that all internal nodes having only leaves in Q are informative, $e(P(Q))$ must be positive. $e(P(Q))$ cannot be 1 since $P(Q)$ cannot be the null tree. So the pruning error cannot reach boundary points -1 or 1. Below we show that both positive and negative cases of $e(Q, P(Q))$ can be realized.

3.4.1 Possible Cases of the Pruning Error

Consider the case of an instance space over three Boolean variables. Further suppose we have a target concept $f()$ and probability distribution $D()$ as shown in Tables 3.2 and 3.3 below. Suppose we take 10 training examples under $D()$. In Table 3.2 and 3.3 below, $n()$ denotes the number of examples for each point x . $f_Q()$ denotes the boolean function of the concept learned by constructing a consistent decision tree from these training examples. By pruning the decision tree Q , we get a pruned decision tree $P(Q)$. $f_{P(Q)}()$ denotes the boolean function of the concept learned from the pruned decision tree. We label each pruned node by the sample labelling rule.

Table 3.2: Positive error case

x	f(x)	D(x)	n(x)	$f_Q(x)$	$f_{P(Q)}(x)$
(1,1,1)	0	.001	0	1	1
(1,1,0)	1	.001	3	1	1
(1,0,1)	1	.001	0	0	0
(1,0,0)	0	.001	2	0	0
(0,1,1)	0	.001	2	0	0
(0,1,0)	1	.001	0	0	0
(0,0,1)	1	.993	2	1	0
(0,0,0)	0	.001	1	0	0

In this decision tree, $r(Q) = 2$ and $r(P(Q)) = 1$.

The pruning error for this case is

$$\begin{aligned}
 & \text{Prob} \{ x : f(x) \neq f_{P(Q)}(x) \} - \text{Prob} \{ x : f(x) \neq f_Q(x) \} \\
 &= (.001 + .001 + .001 + .993) - (.001 + .001 + .001) \\
 &= .993
 \end{aligned}$$

Table 3.3: Negative error case

x	f(x)	D(x)	n(x)	$f_Q(x)$	$f_{P(Q)}(x)$
(1,1,1)	0	.001	0	1	1
(1,1,0)	1	.001	3	1	1
(1,0,1)	1	.001	0	0	0
(1,0,0)	0	.001	2	0	0
(0,1,1)	0	.001	1	0	1
(0,1,0)	1	.993	0	0	1
(0,0,1)	1	.001	3	1	1
(0,0,0)	0	.001	1	0	1

In this decision tree, $r(Q) = 2$ and $r(P(Q)) = 1$.

The pruning error for this case is

$$\begin{aligned} & \text{Prob} \{ x : f(x) \neq f_{P(Q)}(x) \} - \text{Prob} \{ x : f(x) \neq f_Q(x) \} \\ &= (.001 + .001 + .001 + .001) - (.001 + .001 + .993) \\ &= -.991 \end{aligned}$$

3.4.2 The Least Upperbound of the Pruning Error

Here we develop a series of lemmas which we need to determine the least upperbound of the pruning error. This is used to guarantee PAC identification of a learning algorithm. We begin with a formal definition of the least upperbound of the pruning error.

Let W_n^r denote the set of all fully-labeled, reduced, binary trees of rank r over V_n , in which all variables are informative and all internal nodes which have only leaves are informative. Since we are using Findmin() which finds a decision tree $Q \in W_n^r$, we need only consider $Q \in W_n^r$. However, much of the following analysis can be directly applied to other situations.

Definition 3.13: The least upperbound of the pruning error, denoted $\eta_{k,r}$, for given rank k , and target r is defined as follows:

$$\text{Let } \eta_{k,r} = \sup_{Q \in W_n^k} \inf_{P \in \rho(Q)} e(Q, P).$$

For any $Q \in W_n^k$, $\rho(Q)$ is the set of all subtrees of Q of rank r pruned from Q .

For convenience, we define $\eta_r \equiv \eta_{r,r-1}$.

From earlier discussions, we know that $\eta_{k,r}$ can be equal to 1 for an arbitrary distribution D . Since the population distribution is usually unknown for real world situations, it is difficult to characterize $\eta_{k,r}$ further. So, without some restrictions on distribution D , we cannot improve or characterize the upperbound of the pruning error. However, for totally unknown population distributions, we may assume an equally likely distribution (i.e., a uniform distribution), or we may assume that the sampling distribution is equal to the population distribution. Here we start with the assumption of an equally likely distribution D to determine the error of pruning. The pruning error also depends on the labelling rule. Here we use the tree labelling rule. As will be shown, these assumptions will guarantee that the least upperbound of the pruning error is positive and less than or equal to 0.5. In the following we give several results characterizing $\eta_{k,r}$ under these assumptions.

To facilitate our effort to determine $\eta_{k,r}$, we will need to define a Maximal tree. Let $Q(j)$ be a subtree of Q with an internal node j of Q as its root node.

Definition 3.14: A Maximal tree of a reduced decision tree Q with rank r for given n , $n \geq r$, is defined as follows:

- 1) If $n = r$, then Q is a complete binary tree with height n .

- 2) If $n > r$, then for each internal node j ,
- i) if the level of node j is less than $n - r(Q(j))$, then one of the subtrees of $Q(j)$ has rank $r(Q(j))$ and the other subtree has rank $r(Q(j))-1$; or
 - ii) if level of node $j = n - r(Q(j))$, then $Q(j)$ is a complete binary tree with height $r(Q(j))$.

Let M_n^r denote the set of all maximal trees of rank r over V_n .

Based on the above definition, it can be easily seen that the number of nodes are equal for all maximal trees of the same rank for given n , and the number of nodes in a maximal tree increases as the rank increases since a maximal tree $Q \in M_n^r$ can always be found by deleting at least one node from a higher rank maximal tree $Q' \in M_n^{r+1}$.

Under the assumptions of a uniform distribution and the tree labelling rule, Lemma 3.15 and Lemma 3.16 characterizes $\eta_{k,r}$.

Lemma 3.15: When D is a uniform distribution and $\text{Prune}()$ uses the tree labelling rule and is given $Q \in W_n^k$ as input, the least upperbound of the pruning error, $\eta_{k,r}$, is strictly positive and less than 0.5 when $r > 0$. $\eta_{k,r} = 0.5$ only when $r = 0$.

Proof: Let $Q \in W_n^k$. Suppose some internal node "i" in Q is at level m ($m < n$). Further suppose that we prune Q to $P(Q)$ at node "i" leaving "i" a leaf node in $P(Q)$. Let $p("i")$ be the

probability of instances covered by the leaves of the subtree $Q(i)$ in Q . Under a uniform distribution, $p("i") = 2^{n-m}/2^n = (0.5)^m$. Let $e("i")$ be the probability of incorrect classifications in the instances covered by the leaves of subtree $Q(i)$ over the whole instance space.

By the definition of $l_0(i)$ and $l_1(i)$ in Definition 3.8, $l_0(i) + l_1(i) = 2^{n-m}$. Suppose $\alpha_0(i)$ is the number of incorrect classifications of $l_0(i)$ and $\alpha_1(i)$ is the number of incorrect classifications of $l_1(i)$. Then clearly

$$e("i") = \alpha_0(i)/2^n + \alpha_1(i)/2^n$$

where $0 \leq \alpha_0(i) \leq l_0(i)$ and $0 \leq \alpha_1(i) \leq l_1(i)$.

If we prune Q at node $"i"$, then in $P(Q)$ the label of the leaf node $"i"$ is either 0 or 1. If we label the node $"i"$ by the "tree labelling" rule, then node $"i"$ will have the label which covers the larger number of instances. If $l_0(i) \geq l_1(i)$, then the label of the node i is 0 and

$$e("i") = \alpha_0(i)/2^n + l_1(i)/2^n - \alpha_1(i)/2^n.$$

By the definition of the pruning error,

$$e(Q, P(Q)) = e(P(Q)) - e(Q)$$

$$= [\alpha_0(i)/2^n + l_1(i)/2^n - \alpha_1(i)/2^n] - [\alpha_0(i)/2^n + \alpha_1(i)/2^n]$$

$$= l_1(i)/2^n - 2\alpha_1(i)/2^n \leq l_1(i)/2^n \leq (1/2)(0.5)^m$$

since $l_1(i) \leq 2^{n-m-1}$.

Similarly, if $l_0(i) < l_1(i)$, then the label of the node i is 1 and

$$e("i") = \alpha_1(i)/2^n + l_0(i)/2^n - \alpha_0(i)/2^n.$$

$$e(Q, P) = l_0(i)/2^n - 2\alpha_0(i)/2^n \leq l_0(i)/2^n < (1/2)(0.5)^m$$
since $l_0(i) < 2^{n-m-1}$.

In both cases, the pruning error does not exceed half of $p("i")$. In other words, the leaf node "i" in $P(Q)$ correctly classifies at least half of the corresponding instances. Since this is true for all pruned nodes, the pruned tree $P(Q)$ correctly classify at least a half of total instances. Therefore, the pruning error $e(Q, P(Q)) \leq 0.5$.

Further, $l_0(i)$ and $l_1(i)$ are nonzero for any nodes in $Q \in W_n^k$ since all nodes in Q are informative. Hence, $e(Q, P(Q)) > 0$ and $\eta_{k,r} > 0$ for all $k > r \geq 0$.

If $r > 0$, at least one subprocedure of $\text{Prune}()$ stops at step 1 since $\text{Prune}()$ reduces k only in one of its subprocedures and eventually k reduces to less than or equal to r . In that subprocedure, $\text{Prune}()$ does not prune that subtree of Q . So the probability over the instance space for which pruning is needed in Q is strictly less than 1. Hence, $e(Q, P(Q)) < 0.5$ and $\eta_{k,r} < 0.5$.

If $r=0$, then the probability over the instance space for which pruning is needed in Q is equal to 1. Hence, $e(Q, P(Q)) \leq 0.5$ and $\eta_{k,r} = 0.5$ can only be realized when the probability of all the negative examples and that of all the positive examples are equal, and when Q has no error over the instance space, for all $k > r=0$.

□

Lemma 3.16: Let $Q \in W_n^k$ and $P \in \rho(Q)$. The following are true.

- a) The pruning error, $e(Q, P)$, is a nondecreasing function of the amount of pruning, $k-r$.
- b) $\eta_{k,r}$ is attained at a maximal tree $P(Q)$ in M_n^r .
- c) $\eta_{k,r}$ is a nonincreasing function of r .

Proof: a) Suppose that an internal node "a" in Q is at level m ($m < n$) with left child node "b" and right child node "c" each at level $m+1$. Under a uniform distribution, the probability of instances covered by node "a" at level m is $(0.5)^m$. Let $e("i")$ be the probability of an incorrect classification in the instances covered by the leaves of the subtree $Q(i)$.

Suppose $P(Q)$ and $P'(Q)$ are two pruned trees of Q , where $P(Q)$ is found by pruning at nodes "b" and "c" of Q and $P'(Q)$ is found by pruning at node "a" of Q . Further suppose $e("b") = \alpha$ and $e("c") = \gamma$ at level $m+1$. Using the same arguments as in the proof of Lemma 3.15, $0 \leq \alpha, \gamma \leq (0.5)^{m+2}$. Then

$$e("a") = e("b") + e("c") = \alpha + \gamma$$

in $P(Q)$.

But if we prune Q at node "a", the label of the leaf node "a" is either 0 or 1. If the label of "a" is 0, then in $P'(Q)$,

$$\begin{aligned} e("a") &= \alpha + ((0.5)^{m+1} - \gamma) \\ &= (0.5)^{m+1} + \alpha - \gamma. \end{aligned}$$

So, $[e("a") \text{ in } P'(Q)] - [e("a") \text{ in } P(Q)]$

$$= ((0.5)^{m+1} + \alpha - \gamma) - (\alpha + \gamma) = (0.5)^{m+1} - 2\gamma \geq 0$$

since $\gamma \leq (0.5)^{m+2}$.

Similarly, if the label of "a" is 1,

$$\begin{aligned} & [e("a") \text{ in } P'(Q)] - [e("a") \text{ in } P(Q)] \\ &= ((0.5)^{m+1} + \gamma - \alpha) - (\alpha + \gamma) = (0.5)^{m+1} - 2\alpha \geq 0 \end{aligned}$$

since $\alpha \leq (0.5)^{m+2}$.

So, the error of the subtree at node "a" in $P'(Q)$ is greater than or equal to that of $P(Q)$. Since all other nodes are equal except node "a" in $P(Q)$ and $P'(Q)$, $e(P'(Q)) \geq e(P(Q))$. Hence, since $e(Q)$ is equal in both cases, the pruning error, $e(Q, P)$, is a nondecreasing function of the amount of pruning.

b) By the definition of a maximal tree, for given n and r , any nonmaximal tree has less nodes than a maximal tree of equal rank. To find a nonmaximal tree of desired rank by pruning Q , we need to prune more nodes than the amount needed when we find a comparable maximal tree of equal rank. So, the least amount of pruning occurs when we prune Q to a maximal tree $P(Q)$ of rank r . From (a), the least upperbound of the pruning error, $\eta_{k,r}$, is attained at a maximal tree $P(Q)$ in M_n^r .

c) From (b), $\eta_{k,r}$ occurs at a maximal tree. Since a maximal tree with rank $r+1$ has more nodes than a maximal tree with rank r for a given number of attributes n , we need to prune more nodes when we prune a tree with rank k to a tree with rank r than to a tree with rank $r+1$. So, $\eta_{k,r+1} \leq \eta_{k,r}$ for all r .

□

Lemma 3.17 (Characterization of $\eta_{k,r}$):

$$\eta_{k,0} = 0.5 \quad \text{for } k > 0.$$

$$\eta_{k,1} = 0.5 (1 - (0.5)^{k-1}) \quad \text{for } k > 1.$$

$$\eta_{k,2} = 0.5 (1 - (0.5)^{k-2}) - 2(k-2)(0.5)^{k+2} \quad \text{for } k > 2.$$

$$\eta_{k,3} = 0.5 (1 - (0.5)^{k-3}) - (k-3)(k+8)(0.5)^{k+3} \quad \text{for } k > 3.$$

$$\eta_{k,4} = 0.5 (1 - (0.5)^{k-4}) - (k-4) \{ (k^2 + 13k + 84)/3 \} (0.5)^{k+4} \quad \text{for } k > 4.$$

$$\begin{aligned} \eta_{k,r} &= \sum_{m=r+1}^k (0.5)^{k-m+1} \eta_{m,r-1}, \quad k > r. \\ &\leq 0.5 - \{ 1 + (k-r)(0.5)^2 \} (0.5)^{k-r+1} \quad \text{for } k > r > 1. \end{aligned}$$

$$\eta_1 = 1/2 = 0.5. \quad \eta_2 = 1/4 = 0.25. \quad \eta_3 = 3/16 = 0.1875.$$

$$\eta_4 = 5/32 = 0.156. \quad \eta_5 = 35/256 = 0.1367. \quad \eta_6 = 126/1024 = 0.123.$$

$$\eta_7 = 462/4096 = 0.1128. \quad \eta_8 = 1716/16384 = 0.1047.$$

Proof: First we prove that $\eta_{k,1} = 0.5 (1 - (0.5)^{k-1})$. To reduce the rank of a tree to 1, the procedure Prune() will prune each subtree with rank greater than one at each level from level 1 to level $k-1$. So, by Lemma 3.15, the least upperbound of the pruning error for an equally likely distribution will be

$$\begin{aligned} \eta_{k,1} &= (0.5)^2 + (0.5)^3 + \dots + (0.5)^k \\ &= (0.5) (1 - (0.5)^{k-1}). \end{aligned}$$

Now, we prove the recursive relation. With Prune(r, k, Q, S), pruning a maximal tree with rank k to a tree with rank r will result in calls to two subprocedures Prune($r-1, k, Q, S$) and Prune($r, k-1, Q, S$). As the level of pruning goes

down by one level, $\eta_{k,r}$ of the lower level reduces to a half of that of the higher level. So, the pruning errors will have the following relationship.

$$\eta_{k,r} = (0.5)\eta_{k,r-1} + (0.5)\eta_{k-1,r}.$$

If we continue branching the procedure Prune(r,k,Q,S) at each level, we will get

$$\begin{aligned}\eta_{k,r} &= (0.5)\eta_{k,r-1} + (0.5)\{(0.5)\eta_{k-1,r-1} + (0.5)\eta_{k-2,r}\} \\ &= (0.5)\eta_{k,r-1} + (0.5)^2\eta_{k-1,r-1} + \dots + (0.5)^i\eta_{k-i+1,r-1} + \dots \\ &\quad + (0.5)^{k-2}\{(0.5)\eta_{r+2,r-1} + (0.5)\eta_{r+1,r}\}.\end{aligned}$$

Since $\eta_{r+1,r} = (0.5)\eta_{r+1,r-1}$ in a maximal tree,

$$\eta_{k,r} = \sum_{m=r+1}^k (0.5)^{k-m+1} \eta_{m,r-1} \quad k > r$$

where $r+1 \leq m \leq k$.

Here we use mathematical induction to show the upperbound of $\eta_{k,r}$. For $r=2$, the above bound holds because

$$\begin{aligned}\eta_{k,2} &= 0.5 (1 - (0.5)^{k-2}) - 2 (k-2) (0.5)^{k+2} \\ &= 0.5 - (0.5)^{k-3} - (k-2) (0.5)^{k+1} \\ &= 0.5 - \{1 + (k-2) (0.5)^2\} (0.5)^{k-2+1} \quad \text{for } k > r > 1.\end{aligned}$$

Suppose it is true for $r=p-1$, where $p \geq 3$. Then

$$\begin{aligned}\eta_{k,p} &= \sum_{m=p+1}^k (0.5)^{k-m+1} \eta_{m,p-1} \\ &= \sum_{m=p+1}^k (0.5)^{k-m+1} \{0.5 - (0.5)^{m-p+2} - (m-p+1) (0.5)^{m-p+4}\} \\ &= \sum_{m=p+1}^k \{ (0.5)^{k-m+2} - (0.5)^{k-p+3} - (m-p+1) (0.5)^{k-p+4} \} \\ &< 0.5 - (0.5)^{k-p+1} - (k-p) (0.5)^{k-p+3}.\end{aligned}$$

Thus it holds for $r=p$. Therefore,

$$\eta_{k,r} \leq 0.5 - \{1 + (k-r) (0.5)^2\} (0.5)^{k-r+1} \quad \text{for } n > r > 1.$$

□

Lemma 3.18: Let $\eta_{k,r}'$ be the least upperbound of the pruning error using algorithm W-Prune() and the tree labelling rule. When D is a uniform distribution, $\eta_{k,r} \leq \eta_{k,r}'$ for all $k > r \geq 0$.

Proof: In the Lemma 3.15 we assumed only that D is a uniform distribution and the pruning algorithm is the tree labelling rule. Hence, it holds for the procedure W-Prune() also. So, when D is a uniform distribution, $\eta_{k,0} = \eta_{k,0}' = 0.5$ for all $k > 0$.

Now we show that $\eta_{k,1}' = 0.5 (1 - (0.5)^{n-1})$. To reduce the rank of a tree to 1, the procedure W-Prune() will prune each subtree with rank greater than one at each level from level 1 to level $n-1$. So, by Lemma 3.15, the least upperbound of the pruning error for an equally likely distribution will be

$$\begin{aligned}\eta_{k,1}' &= (0.5)^2 + (0.5)^3 + \dots + (0.5)^n \\ &= (0.5) (1 - (0.5)^{n-1}).\end{aligned}$$

Since $k \leq n$, $\eta_{k,1} \leq \eta_{k,1}'$ for all $k > 1$.

Notice that $\eta_{k,1}'$ is independent of k . So, $\eta_{k,1}' = 0.5 (1 - (0.5)^{n-1})$ for all $1 < k \leq n$.

Now we show the resursive relation. With W-Prune(r, k, Q, S), pruning a maximal tree with rank k to a tree with rank r will result in calls to two subprocedures W-Prune(r, k, Q, S) and W-Prune($r-1, k-1, Q, S$). As the level of pruning goes down by one level, $\eta_{k,r}$ of the lower level reduces to a half of that of the higher level. So, the pruning errors will have the following relationship:

$$\eta_{k,r}' = (0.5)\eta_{k,r}' + (0.5)\eta_{k-1,r-1}'.$$

This can be simplified to

$$\eta_{k,r}' = \eta_{k-1,r-1}'.$$

So, the following relationship holds:

$$\eta_{k,r}' = \eta_{k-1,r-1}' = \eta_{k-2,r-2}' = \dots = \eta_{k-(r-1),1}'.$$

Since $\eta_{k,r}$ is a nonincreasing function of r by Lemma 3.16 (c),

$$\eta_{k,r} \leq \eta_{k,1} \leq \eta_{k-(r-1),1}' = \eta_{k,r}' \text{ for all } k > r \geq 1 \text{ follows.}$$

So, when D is a uniform distribution, $\eta_{k,r} \leq \eta_{k,r}'$ for all $k > r \geq 0$. \square

Theorem 3.19: Under the assumption of an equally likely distribution over the instance space, procedure Prune(r, k, Q, S) with the tree labelling rule gives the least upperbound of the pruning error $\eta_{k,r}$ in time $O(2^{k+r} + (en/k)^k)$.

Proof: Case 1 and 3: By reducing the rank of the subtree with rank k to r , the rank of the whole tree will be r . Since $\eta_{k,r}$ is a nondecreasing function of r for fixed k by Lemma 3.16 (c), it gives the least upperbound of the pruning error:

Case 2 and 4: There are only two alternative pruning ways which may give the least upperbound error.

- 1) prune one subtree with rank k to $r-1$ and then prune the other subtree to rank r ; and
- 2) prune one subtree with rank k to r and then prune the other subtree to $r-1$.

By Lemma 3.18, $\eta_{k,r} \leq \eta_{k,r}'$. So, 1) gives the least upperbound error.

Case 5: Since $\eta_r > 0$ for all r by Lemma 3.15, pruning one subtree with rank $k-1$ to a tree with rank r and the other subtree to rank $r-1$ gives the least upperbound of the pruning error.

□

Lemma 3.20 (An upperbound of $\eta_{k,r}$):

$$\begin{aligned} \text{Let } \mu_{n,r} &= 0.5 - (0.5)^{n-r+1} && \text{for } n \geq r=1 \\ &= 0.5 - \{ 1 + (n-r)(0.5)^2 \} (0.5)^{n-r+1} && \text{for } n \geq r > 1. \end{aligned}$$

Then $\eta_{k,r} \leq \mu_{n,r} < 0.5$ for $0 < r \leq k \leq n$.

Proof: In the proof of Lemma 3.17 we have shown that

$$\eta_{k,r} = (0.5)\eta_{k,r-1} + (0.5)\eta_{k-1,r}.$$

Since $\eta_{k,r} \leq \eta_{k,r-1}$ by Lemma 3.16 (c), $\eta_{k,r} \geq \eta_{k-1,r}$. So, $\eta_{k,r}$ is a nondecreasing function of k . And since $r(Q) = k \leq n$ for all reduced binary decision trees, Q , over V_n , $\eta_{k,r} \leq \eta_{n,r}$. By Lemma 3.17, for

$$r=1, \eta_{n,r} = \mu_{n,r} \text{ and for}$$

$$r > 1, \eta_{n,r} \leq \mu_{n,r} = 0.5 - \{ 1 + (n-r)(0.5)^2 \} (0.5)^{n-r+1} \text{ for } n \geq r > 1.$$

For $n-r \geq 0$, the latter term is positive. So, $\mu_{n,r}$ is strictly less than 0.5.

□

3.5. Sample Size Sufficient for PAC Identification

We consider the size of a sample sufficient for PAC identification with pruning in finite domains. Blumer et al. (1987a) have given a lower bound of the sample size required for PAC identification of any consistent learning algorithm. They showed that for N given finite rules in the hypothesis space, any rule agreeing with $m = (1/\epsilon) \ln(N/\delta)$ or more randomly chosen examples has error greater than ϵ only with probability less than δ . This model assumes that there exists a rule in the hypothesis which correctly classifies all the training examples.

The number of possible rules, N , is used as a measure of the complexity of the hypothesis space in PAC-learning. We will use the following lemma when we quantify the hypothesis space for binary decision trees.

Recall that T_n^r denotes the set of all binary decision trees over V_n of rank at most r and F_n^r denotes the set of Boolean functions on X_n that are represented by trees in T_n^r . Then $|F_n^r|$ gives the number of rules in the hypothesis space of decision trees of rank at most r .

Lemma 3.21 (An upperbound on $|F_n^r|$: Ehrenfeucht and Haussler, 1988):

$$|F_n^r| \leq (8n)^{(en/r)r}, \quad \text{for } n \geq r > 0.$$

Consider now what happens if we prune a consistent decision tree. Since pruning a consistent decision tree built with informative variables introduces error over the training samples, the pruned tree P is not consistent with all the training samples. Angluin and Laird (1988) introduced into PAC-learning a model of random errors, or "noise", in the sampling. They assumed that it is possible to draw examples from the relevant distribution D without error, but that the process of determining and reporting whether the example is positive or negative is subject to independent random mistakes with some unknown probability $\eta < 0.5$. They called this process "the classification noise process" and developed a result which guarantees PAC-identification for the process. Their result is as follows.

Lemma 3.22 (Angluin and Laird, 1988):

For any finite set of N rules,
if we draw a sequence of

$$m \geq \left\lceil \frac{2}{\epsilon^2(1-2\eta_b)^2} \right\rceil \ln(2N/\delta)$$

random examples for target concept f from an arbitrary probability distribution D with classification noise η (an upperbound is $\eta_b < 1/2$) and find any hypothesis h that minimizes the probability of misclassification in the examples, then

$$\text{Prob}\{ d(h,f) \geq \epsilon \} \leq \delta.$$

Our model does not assume classification noise. However, the pruned tree is known to be in an hypothesis space where no function h is exactly equivalent to the target function f . Hence, the concepts we learn with pruning appear as if they are subject to a classification error.

Since we defined the pruning error over the instance space, the pruning error can be considered as a classification noise in the random examples.

A direct application of Lemma 3.22 with our results gives our main result.

Theorem 3.23: For any $n \geq r > 0$, any target function $f \in F_n^p$, $p \leq n$, an equally likely probability distribution D on X_n and any $0 < \epsilon, \delta < 1$, given a sample S derived from a sequence of

$$m \geq \left\lceil \frac{2}{\{\epsilon^2(1-2\mu_{n,r})^2\}} \left\{ (en/r)^r \ln(8n) + \ln(2/\delta) \right\} \right\rceil$$

random examples of f chosen independently according to D , with probability $1-\delta$, Findmin(S) and Prune(r, k, Q, S) using tree labelling produces a hypothesis $h \in F_n^r$ that has error at most ϵ .

Proof: Since we defined the pruning error over the whole instance space, the pruning error for the consistent decision tree can be considered as a classification noise over the random sample without violating the Angluin and Laird framework. The number of rules N in a decision tree of rank at most r is bounded above by Lemma 3.21. Findmin(S) produces a consistent decision tree and Prune(r, k, Q, S) minimizes the

probability of misclassification over the whole instance space with confidence probability 1. Lemma 3.22 holds for any probability distribution D with $\eta_b < 0.5$ and for any algorithm which minimizes the misclassification over the whole instance space with confidence probability $1 - \delta$ which is less than 1. Letting $\eta_b = \mu_{n,r}$, and noting that $\mu_{n,r}$ is less than 0.5 for all $n \geq r > 0$, produces the desired result.

□

Combined with the result of Findmin(S) in Theorem 3.6 and our analysis of Prune(r,k,Q,S) in Theorem 3.19, Theorem 3.23 shows that the decision tree with rank at most r on n variables can be learned with pruning with accuracy $1 - \epsilon$ and confidence $1 - \delta$ in time polynomial in $1/\epsilon$, $1/\delta$, $1/(1 - 2\mu_{n,r})$ and n for fixed rank r , allowing one unit of time to draw each random examples. Thus pruning has increased the number of examples we must obtain, but still retains its polynomial sampling characteristic. If the rank of the induced decision tree, k , can be estimated probabilistically, Theorem 3.23 can be tightened by replacing $\mu_{n,r}$ by $\mu_{k,r}$.

The above sample sizes in Theorem 3.23 can be reduced by using Laird's (1987) improved bound on learning from noisy examples. Theorem 3.24 presents this bound.

Theorem 3.24: Assume $0 < \epsilon, \delta < 1/2$. For any $n \geq r > 0$, any target function $f \in F_n^p$, $p \leq n$, an equally likely probability

distribution D on X_n and any $0 < \varepsilon, \delta < 1/2$, given a sample S derived from a sequence of

$$m \geq \left\lceil \frac{1}{\varepsilon (1 - \exp(-(0.5)(1 - 2\mu_{n,r})^2))} \right\rceil \left\{ (en/r)^r \ln(8n) + \ln(1/\delta) \right\}$$

random examples of f chosen independently according to D , with probability $1 - \delta$, Findmin(S) and Prune(r, k, Q, S) using tree labelling produces a hypothesis $h \in F_n^r$ that has error at most ε .

In this following we illustrate findings by using the loan default problem given in Messier and Hansen (1988). For presentation ease, we limit the instance space to 6 financial ratios. We are to find rules predicting loan default of a company using the following list of attributes. Here we express all the variables as binary variables with values low and high. These split come from Messier and Hansen (1988).

Attributes	low	high
Current Ratio	< 1.912	≥ 1.912
Long-term Debt/Net Worth	$< .486$	$\geq .486$
Lo Long-term Debt/Net Worth	$< .046$	$\geq .046$ and $< .486$
Working Capital/Sales	$< .222$	$\geq .222$
Net Income/Total Assets	$< .100$	$\geq .100$
Net Income/Sales	$< .010$	$\geq .010$

Suppose the hypothesis space H is decision trees. Then, the number of rules in H is $|H| = 2^{64} = 1.8447 \times 10^{19} = N$.

Here we illustrate how decision tree construction and pruning algorithms work. First we construct a consistent decision tree with Findmin(S), where S is in Table 3.4. (This is the same sequence of examples used in Messier and Hansen 1988.)

Let 0 = low, and 1 = high. Algorithm Findmin(S) begins with Find(S,0), and returns "none" since the examples in S are not in the same class.

Now the algorithm calls Find(S,1). "Current Ratio" is an informative variable. In Step 3.a it calls two subprocedures, Find(S_0^v ,0) and Find(S_1^v ,0). Find(S_0^v ,0) is successful but Find(S_1^v ,0) is not successful. In Step 3.c it calls Find(S_1^v ,1). We see that any set of examples in each branch of each variable (attribute) are not in the same class. Hence Find(S_1^v ,1) returns "none". In a similar procedure we will see that Find(S,1) returns "none" since for any other informative variables than "Current Ratio", S_0^v is not in the same class and S_1^v is not in the same class, respectively.

Now Findmin(S) calls Find(S,2). "Current Ratio" is an informative variable. In Step 3.a it calls two subprocedures, Find(S_0^v ,1) and Find(S_1^v ,1). Find(S_0^v ,1) is successful but Find(S_1^v ,1) is not successful. In Step 3.c it calls Find(S_1^v ,2). Now let $S = S_1^v$. "Long-term Debt/Net Worth" is an informative variable in this subsample. In Step 3.a it calls two subprocedures, Find(S_0^v ,1) and Find(S_1^v ,1). We see that both subprocedures are successful. Hence Find(S,2)

Table 3.4: Sample S for Loan Default problem

Values of Attributes*						Class**
(1)	(2)	(3)	(4)	(5)	(6)	
2.767	0.286	0.286	0.383	0.045	0.037	N
1.540	0.985	n/a***	1.171	0.017	0.031	D
1.680	0.203	0.203	0.169	0.023	0.014	D
1.548	0.334	0.334	0.211	0.034	0.023	D
3.460	0.415	0.415	0.393	0.052	0.037	N
2.539	0.656	n/a	0.440	0.045	0.524	D
3.085	0.102	0.102	0.251	0.136	0.066	N
1.762	0.375	0.375	0.175	0.107	0.069	D
5.094	0.486	n/a	0.469	0.132	0.153	D
2.521	0.239	0.239	0.223	0.085	0.048	N
1.770	0.088	0.088	0.223	0.033	0.031	D
2.826	0.485	0.485	0.288	0.038	0.033	N
1.269	1.133	n/a	0.072	0.052	0.130	D
4.042	0.472	0.472	0.697	0.049	0.048	N
2.246	0.972	n/a	0.236	0.036	0.035	D
1.879	0.879	n/a	0.178	0.043	0.028	D
2.582	0	0	0.445	0.143	0.154	N
2.338	0.043	0.043	0.378	0.055	0.054	D
2.159	0.239	0.239	0.113	0.019	0.005	N
2.005	0.748	n/a	0.170	0.089	0.530	N
2.156	1.089	n/a	0.204	0.033	0.021	N
1.940	1.237	n/a	0.266	0.018	0.013	D
1.977	0.161	0.161	0.211	0.066	0.062	N
1.920	5.474	n/a	0.149	0.015	0.011	N
2.330	0.049	0.049	0.276	0.052	0.041	N
1.930	0.621	n/a	0.145	0.024	0.009	D
1.904	0.042	0.042	0.158	0.076	0.054	D
1.788	0.401	0.401	0.202	-0.063	-0.043	D
2.419	0.074	0.074	0.161	0.127	0.062	N
2.080	0.932	n/a	0.208	0.054	0.059	N
1.341	12.780	n/a	0.091	-0.046	-0.051	D
2.616	0.185	0.185	0.235	0.090	0.115	N

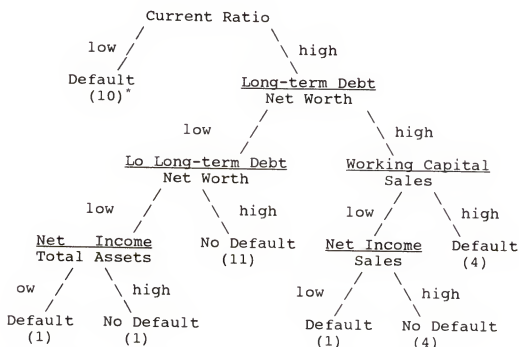
*: Attributes in each column have following names.

- (1) Current Ratio (2) Long-term Debt/Net Worth
 (3) Lo Long-term Debt/Net Worth (4) Working Capital/Sales
 (5) Net Income/Total Assets (6) Net Income/Sales

** : (N) Not Default, (D) Default

*** : (n/a) not applicable

returns a consistent decision tree of rank two in Figure 3.1. Thus Findmin(S) returns a minimal rank decision tree of rank two consistent with sample S.

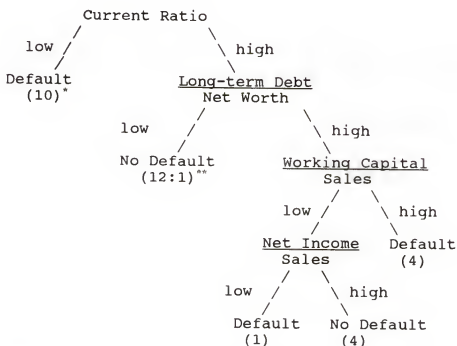


*: The number of examples in each terminal node.

Figure 3.1: A decision tree predicting loan default.

Suppose we want a decision tree of rank one. We prune the above decision tree in Figure 3.1 with Prune(r, k, Q, S). Here $r = 1$, $k = 2$, Q = the decision tree in Figure 3.1, and S is the sample used in Messier and Hansen (1988). Since Q is in Case 3, Prune($1, 2, Q, S$) calls one subprocedure, Prune($1, 2, Q_1, S_1$) with $Q = Q_1$ and $S = S_1$. Here, the procedure calls two subprocedures, Prune($0, 1, Q_0, S_0$) and Prune($1, 1, Q_1, S_1$), since the new Q is in Case 5. By Step 2, Prune($0, 1, Q_0, S_0$)

returns "Q = No Default" by Tree Labelling. By Step 1, $\text{Prune}(1,1,Q_1,S_1)$ returns Q_1 . Hence $\text{Prune}(1,2,Q,S)$ returns a pruned decision tree of rank one in Figure 3.2.



*: The number of examples in each terminal node.

** : 12 examples are in No Default, One example is in Default.

Figure 3.2: A pruned decision tree predicting loan default.

Now we give sample sizes sufficient for learning a decision tree with pruning. We use the above decision tree of rank one pruned from an induced decision tree of rank two. Here $n = 6$, $k = 2$, and $r = 1$. Then, by Theorem 3.23, the number of examples, m , sufficient for learning a decision tree of rank one is

for $\epsilon = 0.5$ and $\delta = 0.01$, $m = 560,609$, and

for $\epsilon = 0.1$ and $\delta = 0.01$, $m = 14,015,240$.

The above sample sizes are very large because of the loose bounds of $\mu_{n,r}$ and F_n^ε . Since $k = 2$, $\mu_{6,1}$ can be substituted by the least upper bound of the pruning error, $\eta_{2,1} = 0.25$. In fact, $|F_n^\varepsilon| \leq N = 2^{64}$ for any value of r . With these tighter bounds the number of examples, m , sufficient for learning reduces to

for $\varepsilon = 0.5$ and $\delta = 0.01$, $m = 6,356$ and

for $\varepsilon = 0.1$ and $\delta = 0.01$, $m = 158,895$.

By using Theorem 3.24 and tighter bounds for $\mu_{n,r}$ and F_n^ε , the sufficient sample size could be reduced as follows:

For $\varepsilon = 0.499$ and $\delta = 0.01$, $m = 833$,

for $\varepsilon = 0.2$ and $\delta = 0.01$, $m = 2,084$, and

for $\varepsilon = 0.1$ and $\delta = 0.01$, $m = 4,168$ are obtained as the number of examples sufficient for learning.

Hence, the learned concept (that is, a pruned decision tree of rank at most one) h has error less than 10% with probability greater than 99% after 4,168 random independent examples.

3.6 Other Pruning Rules

In Section 3.4, we determined the least upperbound of the pruning error with the assumption of an equally likely distribution and the tree labelling rule. Here we give additional insight into the pruning error by considering some problematic cases.

In Section 3.4, we saw that sample labelling under a nonuniform distribution could give a pruning error greater than 0.5. Now consider the sample labelling rule under the assumption of an equally likely distribution. Table 3.5 presents an example which shows the least upperbound of the pruning error may be greater than 0.5. We use the same notation as in Tables 3.2 and 3.3 of Section 3.4.

Table 3.5: Sample labelling with a uniform distribution

x	$f(x)$	$D(x)$	$n(x)$	$f_Q(x)$	$f_{P(Q)}(x)$
(1,1,1,1)	1	1/16	1	1	1
(1,1,1,0)	0	1/16	1	0	0
(1,1,0,1)	1	1/16	1	1	0
(1,1,0,0)	0	1/16	2	0	0
(1,0,1,1)	1	1/16	1	1	0
(1,0,1,0)	1	1/16	0	1	0
(1,0,0,1)	1	1/16	1	1	0
(1,0,0,0)	0	1/16	3	0	0
(0,1,1,1)	1	1/16	2	1	0
(0,1,1,0)	1	1/16	0	1	0
(0,1,0,1)	1	1/16	0	1	0
(0,1,0,0)	1	1/16	0	1	0
(0,0,1,1)	1	1/16	1	1	0
(0,0,1,0)	1	1/16	0	1	0
(0,0,0,1)	1	1/16	1	1	0
(0,0,0,0)	0	1/16	6	0	0

In this decision tree, $r(Q) = 2$ and $r(P(Q)) = 1$. The pruning error in this case is

$$\begin{aligned} & \text{Prob} \{ x : f(x) \neq f_{P(Q)}(x) \} - \text{Prob} \{ x : f(x) \neq f_Q(x) \} \\ &= 7/16 + 3/16 + 1/16 = 11/16 > 0.5. \end{aligned}$$

We now turn to labelling rules that are not deterministic. In such methods we may use some probabilistic labelling procedure where the pruned node has a probability of the target function $f(x) = 0$ and $f(x) = 1$ in proportion to a corresponding sample or instance. As we will see, the pruning error may exceed 0.5 for both sample labelling and tree labelling. Below we define two nondeterministic labeling methods.

Definition 3.25 (Nondeterministic Labelling): Let i be an internal node in a decision tree Q , and $Q(i)$ be a subtree of Q such that node i is the root of the tree $Q(i)$.

1. Sample labelling:

Let $s(i)$ be a subset of the sample S used to construct Q from its root to node i .

Let $s_0(i)$ denote the number of negative examples in $s(i)$ and $s_1(i)$ denote the number of positive examples in $s(i)$. If we prune Q at i , then label i as

$$\begin{cases} 0 & \text{with probability } s_0(i) / (s_0(i) + s_1(i)) \text{ and} \\ 1 & \text{with probability } s_1(i) / (s_0(i) + s_1(i)). \end{cases}$$

2. Tree labelling:

Let v_{i_1}, \dots, v_{i_p} be the nodes in the path from the root to the parent of node i in Q . And let a_{i_1}, \dots, a_{i_p} be the labels of each node, where i_1, \dots, i_p are p distinct indices of $\{1, \dots, n\}$.

Let $l_0(i) = |\{x: f_Q(x)=0\}|$, where $x \in X_n$ such that a_{i_1}, \dots, a_{i_p} are the same.

Let $l_1(i) = |\{x: f_Q(x)=1\}|$, where $x \in X_n$ such that a_{i_1}, \dots, a_{i_p} are the same. If we prune Q at i , then label i as

$$\begin{cases} 0 & \text{with probability } l_0(i) / (l_0(i) + l_1(i)) \text{ and} \\ 1 & \text{with probability } l_1(i) / (l_0(i) + l_1(i)). \end{cases}$$

By using the case shown in Table 3.5, we show that the pruning error may exceed 0.5 with both nondeterministic labelling methods. If we use nondeterministic sample labelling, then the label of the pruned node may have a 0 or 1 label with the corresponding probabilities defined in Definition 3.24. Since $s_0()$ are positive for all pruned nodes of $P(Q)$ in the case shown in Table 3.5, all pruned nodes may have 0 labels. In that case the pruning error is

$$\begin{aligned} &= 0.5(0.875-0) + 0.25(0.75-0) + 0.125(0.5-0) \\ &= 0.6875 > 0.5. \end{aligned}$$

Similarly, since $l_0()$ are positive for all pruned nodes of $P(Q)$, all pruned nodes may have 0 labels. In that case the pruning error for tree labelling is equal to that for sample labelling calculated above. So, even for an equally likely

distribution D , the least upperbound of the pruning error may exceed 0.5 under nondeterministic labelling.

However, if we use the nondeterministic tree labelling rule, the upperbound of the average pruning error is less than or equal to 0.5 under a uniform distribution. The following lemma shows this.

Lemma 3.26: When D is a uniform distribution and $\text{Prune}()$ uses the nondeterministic tree labelling rule, the upperbound of the average pruning error, $m_{k,r}$, is less than or equal to 0.5 for all $k > r \geq 0$.

Proof: Let $Q \in W_n^k$. Suppose some internal node "i" in Q is at level m ($m < n$). Further suppose that we prune Q to $P(Q)$ at node "i" leaving "i" a leaf node in $P(Q)$. Let $p("i")$ be the probability of instances covered by the leaves of the subtree $Q(i)$ in Q . Under a uniform distribution, $p("i") = 2^{n-m}/2^n = (0.5)^m$. Let $e("i")$ be the probability of incorrect classifications of the instances covered by the leaves of subtree $Q(i)$ over the whole instance space.

By the definition of $l_0(i)$ and $l_1(i)$ in Definition 3.8, $l_0(i) + l_1(i) = 2^{n-m}$. Suppose $\alpha_0(i)$ is the number of incorrect classifications of $l_0(i)$ and $\alpha_1(i)$ is the number of incorrect classifications of $l_1(i)$. Then clearly

$$e("i") = (0.5)^m (\alpha_0(i) + \alpha_1(i)) / (l_0(i) + l_1(i))$$

where $0 \leq \alpha_0(i) \leq l_0(i)$ and $0 \leq \alpha_1(i) \leq l_1(i)$.

If we prune Q at node "i", then in $P(Q)$ the label of the leaf "i" is

$$\begin{cases} 0 & \text{with probability } l_0(i) / (l_0(i) + l_1(i)) \text{ and} \\ 1 & \text{with probability } l_1(i) / (l_0(i) + l_1(i)). \end{cases}$$

Below we suppress (i) for clearer reading. If the label of "i" is 0, then

$$e("i") = (0.5)^m (1 - (l_0 - \alpha_0 + \alpha_1) / (l_0 + l_1))$$

since the true probability of a 0 label is $(l_0 - \alpha_0 + \alpha_1) / (l_0 + l_1)$.

By the definition of the pruning error,

$$\begin{aligned} e(Q, P(Q)) &= e(P(Q)) - e(Q) \\ &= (0.5)^m [((l_1 + \alpha_0 - \alpha_1) - (\alpha_0 + \alpha_1)) / (l_0 + l_1)] \\ &= (0.5)^m (l_1 - 2\alpha_1) / (l_0 + l_1). \end{aligned}$$

Similarly, if the label of node "i" is 1, then

$$e(Q, P(Q)) = (0.5)^m (l_0 - 2\alpha_0) / (l_0 + l_1).$$

So, the average pruning error is

$$\begin{aligned} &= (l_0 / (l_0 + l_1)) * (0.5)^m (l_1 - 2\alpha_1) / (l_0 + l_1) \\ &+ (l_1 / (l_0 + l_1)) * (0.5)^m (l_0 - 2\alpha_0) / (l_0 + l_1) \\ &= (0.5)^m (2l_0l_1 - 2l_0\alpha_1 - 2l_1\alpha_0) / (l_0 + l_1)^2 \\ &\leq (0.5)^m (2l_0l_1) / (l_0 + l_1)^2 \leq (1/2)(0.5)^m \end{aligned}$$

since $4l_0l_1 \leq (l_0 + l_1)^2$ and $\alpha_0, \alpha_1 \geq 0$.

Since the above bound holds for all pruned nodes, the upperbound of the average pruning error, $m_{k,r}$, is less than or equal to 0.5.

□

So, the nondeterministic tree labelling rule can hedge the risks of biased nonrandom selections of examples by guaranteeing that the average pruning error does not exceed 0.5 regardless of the sampling distribution.

3.7 Chapter Summary

Empirical results have shown that pruning can improve the accuracy of an induced decision tree. It also leads to more concise rules. Here we provide a pruning algorithm based on the rank of a decision tree. A bound on the error due to pruning by the rank of a decision tree is determined under the assumptions of an equally likely distribution of the instance space and a deterministic tree labelling rule. This bound is then used with recent results in learning theory to determine a sample size sufficient for PAC identification of decision trees with pruning. We also discuss other pruning rules and their effects on the error due to pruning. With nondeterministic tree labelling rule we show that the upperbound of the average pruning error is less than or equal to 0.5 under an equally likely distribution.

Future work will be needed to determine the pruning error under more general assumptions on the distribution over the instance space. Also, Theorem 3.23 can be tightened if an a prior estimate, k , of the rank of the induced decision tree can be determined. If so, $\mu_{n,T}$ can be replaced by $\mu_{k,T}$.

Here we take the rank of a tree as a conciseness measure of a decision tree. Future work will be needed to assess the effect of pruning under other conciseness criteria.

CHAPTER 4 THE ACCURACY OF A PRUNED DECISION TREE

4.1 Introduction

In the previous chapter we provided a bound on the training sample size to guarantee PAC learning of decision trees with pruning. In many learning environments it is often not possible to obtain a sample large enough to guarantee PAC learning. In such cases it is of value to obtain a posterior evaluation of the accuracy of a pruned decision tree. For this purpose, a set of examples, called the test set, is employed. Here it is important to ensure that the test set can be considered as independent of the sample used in constructing the decision tree, and drawn from the same distribution. So, a test set is often a holdout sample randomly removed from original cases available for training. Often the holdout sample is $1/3$ or $1/2$ of the original sample available for training though there is no theoretical justification for that ratio.

Tsai and Koehler (1991) give an excellent result on the posterior measurement for the error and confidence parameters (ϵ, δ) . They assume that there is a rule which is consistent with all training samples. The use of their result for pruned

trees is, however, inappropriate since pruning a decision tree always leads to an inconsistent decision rule. In this chapter we present some methods for error and confidence parameter estimation for a pruned decision tree.

4.2 The Estimation of Error of a Pruned Decision Tree

In this section we present three methods for error and confidence parameter estimation for the pruned decision tree. First recall the basic definitions given in Chapters 2 and 3.

Let X be the instance space of interest. The target concept f maps X into $\{0,1\}$. Similarly, for any other concept h , we have

$$h: X \rightarrow \{0,1\}.$$

The error, $d(h,f)$, of a learned concept h is the probability of the instances incorrectly classified by h . That is,

$$d(h,f) = \text{Prob}\{ x \in X: h(x) \neq f(x) \}.$$

$\text{Prob}\{\}$ is determined by an arbitrary sampling distribution, D , over X . Sampling is assumed to be with replacement with samples drawn independently.

Let $\theta = d(h,f)$, where h is a pruned tree $P(Q)$, and f is the target concept (the correct decision tree). The following result gives a posterior estimate of the accuracy of a pruned tree $P(Q)$, determined using the independent test sample of size m .

4.2.1 When No Prior Information Exists

We may assume a uniform distribution of error when there is no reliable information for the distribution of error. The following result gives a posterior estimate of the accuracy of a pruned decision tree determined using a test sample assuming a uniform prior (Raiffa and Schlaifer, 1961; Winkler 1972).

Theorem 4.0: (Posterior Estimate of the Confidence Parameter)

Given b misclassifications (i.e. failures) in a test sample of size m , the posterior estimate of the binomial parameter, θ , using a uniform prior is

$$Prob\{\theta \geq \epsilon \mid b, m\} = \sum_{k=0}^b C_k^{m+1} \epsilon^k (1-\epsilon)^{m+1-k} \quad \text{for } \epsilon \leq 0.5$$

and

$$Prob\{\theta \geq \epsilon \mid b, m\} = \sum_{k=m+1-b}^{m+1} C_k^{m+1} (1-\epsilon)^k \epsilon^{m+1-k} \quad \text{for } \epsilon \geq 0.5.$$

Here C_k^m is the number of ways of unordered sampling of size k out of m .

The terms in Theorem 4.0 are easily computed using the Incomplete Beta distribution and methods given in Abramowitz and Segun (1968) or approximated using methods given by Peizer and Pratt (1968). As indicated by Tsai and Koehler (1991), a uniform prior is often inappropriate since the error often has high probability in a certain range.

In the next subsection we develop probabilistic estimates of the error that are based on the assumption of a Beta prior distribution.

4.2.2 Using the Information of the Training Sample

We may assume a Beta prior to obtain a parameter set consistent with the training information. Then, using the test sample, we can determine a posterior estimate for the error. To derive various estimates we will need Hoeffding's inequality.

Lemma 4.1: (Hoeffding Inequalities: Hoeffding, 1963)

Let x_1, x_2, \dots, x_n be independent random variables with $0 \leq x_i \leq 1$ and $E[x_i] = \mu$, for $i = 1, 2, \dots, n$,

$$\bar{x} = (\sum_{i=1}^n x_i) / n.$$

Then,

$$\text{Prob}(\bar{x} - \mu \geq c) \leq \exp(-2nc^2)$$

and

$$\text{Prob}(\mu - \bar{x} \geq c) \leq \exp(-2nc^2).$$

Suppose that there are b misclassifications out of the training sample of size m . By Hoeffding's inequality, the following bound holds.

Lemma 4.2:

For any $b/m \leq \epsilon$,

$$\text{Prob}(\theta \geq \epsilon) \leq \exp(-2(\epsilon - b/m)^2 m).$$

Proof: Let $x_i = 1$ if a pruned decision tree misclassifies the i^{th} example and $x_i = 0$ otherwise. The probability of misclassification θ is the expected value $E[x_i]$. Since we have b misclassifications in the independent sample, $\bar{x} = b/m$.

$\text{Prob}(\theta \geq \epsilon) = \text{Prob}(\theta - b/m \geq \epsilon - b/m) \leq \exp(-2(\epsilon - b/m)^2 m)$ holds by Lemma 4.1.

□

Let Z^+ be the set of positive integers and let $S(b,m)$ be the set $\{(p,q): p,q \in Z^+\}$ where, for all $b/m \leq \epsilon < 1$, $I_\epsilon(p,q)$ is an Incomplete Beta distribution that is consistent with the bound in Lemma 4.2. That is, $S(b,m)$ is the set of integer parameters for Incomplete Beta distributions that are consistent with the bound in Lemma 4.2. In the following we develop a characterization of $S(b,m)$.

First, Lemma 4.3, 4.4 and 4.5 develop necessary conditions for consistent (p,q) set. Suppose that there are b misclassifications out of the training sample of size m .

Lemma 4.3: For $p,q \in Z^+$ and $b/m \leq \epsilon < 1$, where $1 \leq b \leq m-1$, then

$$(p,q) \in S(b,m)$$

only if $p + q \geq 1 + 4m(1-\epsilon)(\epsilon-b/m)$, where ϵ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m$.

Proof: By Lemma 4.2, any consistent Beta prior must satisfy

$$\text{Prob}(\theta \geq \epsilon) = 1 - I_{\epsilon}(p, q) \leq \exp(-2(\epsilon-b/m)^2 m)$$

for all $b/m \leq \epsilon < 1$. Here $I_{\epsilon}(p, q)$ is the Incomplete Beta which gives the probability of a value less than or equal to ϵ . Since p and q are integers, the above inequality can be written with a Binomial distribution as follows:

$$\sum_{k=0}^{p+q-1} C_k^{q+p-1} \epsilon^k (1-\epsilon)^{p+q-1-k} \leq e^{-2(\epsilon-b/m)^2 m}.$$

By rearranging terms, the following inequality holds:

$$\sum_{k=0}^{p+q-1} C_k^{q+p-1} e^{k \ln \epsilon + (p+q-1-k) \ln(1-\epsilon) + 2(\epsilon-b/m)^2 m} \leq 1.$$

Then, for $k=0$,

$$(p+q-1) \ln(1-\epsilon) + 2(\epsilon-b/m)^2 m \leq 0.$$

That is,

$$p+q-1 \geq -2(\epsilon-b/m)^2 m / \ln(1-\epsilon).$$

Let $f(\epsilon) = -2(\epsilon-b/m)^2 m / \ln(1-\epsilon)$.

Since $f(\epsilon)$ is continuous and concave on an interval where its maximum occurs ($f''(\epsilon) < 0$ at unique ϵ satisfying $f'(\epsilon)=0$ if $\ln(1-\epsilon) < 0.5$), by taking its first derivative and setting it equal to 0, we get ϵ giving the maximum value of $f(\epsilon)$. I.e., $f'(\epsilon) = 0$ reduces to

$$-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m.$$

So,

$$p+q \geq 1 - 2(\epsilon-b/m)^2 m / \ln(1-\epsilon) = 1 + 4m(1-\epsilon)(\epsilon-b/m),$$

where ϵ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m$, is obtained as a necessary condition.

□

Below, Lemma 4.4 gives a relationship between the consistent Beta priors in $S(b,m)$.

Lemma 4.4: (Tsai and Koehler 1991) Assume $p, q \in Z^+$. If

$$(p,q) \in S(b,m)$$

then

$$(t,q) \in S(b,m) \quad 1 \leq t \leq p$$

$$(p,t) \in S(b,m) \quad t \geq q.$$

Lemma 4.5 gives a necessary condition for parameter q in $S(b,m)$.

Lemma 4.5: For $p,q \in Z^+$ and $b/m \leq \epsilon < 1$, where $1 \leq b \leq m-1$, then

$$(p,q) \in S(b,m)$$

only if $q \geq 4m(1-\epsilon)(\epsilon-b/m)$, where ϵ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m$.

Let q^* be the minimum q satisfying the above condition. Then $(1,q^*)$ is a consistent Beta prior.

Proof: Let $p = 1$. Then

$$\text{Prob}\{\theta \geq \epsilon\} = 1 - I_\epsilon(1,q) = (1-\epsilon)^q \leq \exp(-2(\epsilon-b/m)^2m)$$

By taking logarithms, we get

$$q \ln(1-\epsilon) \leq -2(\epsilon-b/m)^2 m.$$

By a similar reasoning as in the proof of Lemma 4.3, we get

$$q \geq 4m(1-\epsilon)(\epsilon-b/m), \text{ where } \epsilon \text{ is the root of the equation} \\ -2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m.$$

Let q^* be the minimum q satisfying the above condition. Then $(1, q^*)$ is a consistent Beta prior since it satisfies the necessary condition in Lemma 4.3 and the probability bound holds for any ϵ , where $b/m \leq \epsilon < 1$.

Suppose, by contradiction, $(1+k, q^*-k)$, where $k \in \mathbb{Z}^+$, is a consistent Beta prior. Note that this Beta prior satisfies a necessary condition in Lemma 4.3. Then by Lemma 4.4, $(1, q^*-k)$ is also a consistent Beta prior. So, q^* is not the minimum q , given $p=1$, consistent with above bound. Hence we reach a contradiction. So, $(1+k, q^*-k)$ is not a consistent Beta prior for any $k \in \mathbb{Z}^+$.

Since for any $q < q^*$, the minimal possible p is not consistent, consistent q should be greater than or equal to q^* for any $p \in \mathbb{Z}^+$.

Hence, $(p, q) \in S(b, m)$

only if $q \geq 4m(1-\epsilon)(\epsilon-b/m)$,

where ϵ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m$.

□

Suppose (p, q) is a consistent Beta prior. Further suppose there are b_2 misclassifications in a test sample of

size m_2 . Then the posterior distribution will have parameters $(p+b_2, q+m_2-b_2)$. The worst possible confidence factor, δ , is

$$\delta = \text{Sup } 1 - I_\epsilon(p+b_2, q+m_2-b_2)$$

subject to

$$(p, q) \in S(b, m)$$

$$p, q \in Z^+$$

By Lemma 4.4, we know that the term

$$I_\epsilon(p+b_2, q+m_2-b_2)$$

decreases with increases in p and decreases in q . So, the supremum may be attained at (p, q) , where p is the maximal p among all possible p , and q is the minimal q among all possible q . However, as p increases, the consistent minimal q increases (more strictly, it does not decrease) by Lemma 4.4. So, in most cases, we cannot obtain such an ideal (maximal p , minimal q) set. Therefore, we need to get a more detailed relationship for the consistent parameter set.

Below, Lemma 4.6 gives a relationship between a consistent Beta prior (p, q) and (p', q') , where $p' > p$, and $q' > q$. Here we will see that the decision of determining the parameter set, giving the higher confidence factor, δ , depends on the value of ϵ .

Lemma 4.6: Assume $k \in Z^+$. Then the following are true.

a) Suppose (p,q) and $(p+k,q+1)$ are consistent Beta priors, where $k \in \mathbb{Z}^+$.

If $q/(p+q) \geq b/m$, then $(p+k,q+1)$ gives a higher confidence factor for $b/m \leq \epsilon < \alpha$, and (p,q) gives a higher confidence factor for $\alpha < \epsilon < 1$, where $b/m \leq \alpha < 1$. As k increases, α increases.

If $q/(p+q) < b/m$, and $k=1$, then (p,q) gives a higher confidence factor than $(p+1,q+1)$ for all ϵ , $b/m \leq \epsilon < 1$.

In no case is $\alpha \geq 1$.

b) Suppose (p,q) and $(p+1,q+k)$ are consistent Beta priors, where $k \in \mathbb{Z}^+$.

If $q/(p+q) \geq b/m$, then $(p+1,q+k)$ gives a higher confidence factor for $b/m \leq \epsilon < \beta$, (p,q) gives higher confidence factor for $\beta < \epsilon < 1$, where $b/m \leq \beta < 1$. As k increases, β decreases.

For example, for $k=2$, and p, q are large enough compared to 1, and p/q is close to $1/2$, then β is close to $1/3$. As p/q increases, β also decreases.

Proof: a) Suppose $k=1$. From Abramowitz and Segun (1968), by combining formula 26.5.15 and 26.5.16, we get

$$g(k) = g(1) = I_c(p,q) - I_c(p+1,q+1) = K * (1 - (p+q)/q \epsilon),$$

where $K = \epsilon^p(1-\epsilon)^q \Gamma(p+q) / \Gamma(p+1)\Gamma(q)$.

Since K is positive, $g(1) = I_c(p,q) - I_c(p+1,q+1) > 0$ for $b/m \leq \epsilon < q/(p+q)$, and $g(1) < 0$ for $q/(p+q) < \epsilon < 1$. So, if $q/(p+q) > b/m$, then $(p+1,q+1)$ gives a higher confidence factor

for $b/m \leq \varepsilon < q/(p+q)$, (p,q) gives a higher confidence factor for $q/(p+q) < \varepsilon < 1$. If $q/(p+q) < b/m$, then (p,q) gives a higher confidence factor for all ε , $b/m \leq \varepsilon < 1$.

Now we can derive a general formula for $g(k)$ by substituting $I_\varepsilon(p+(k-1),q+1)$ for $I_\varepsilon(p+k,q+1)$ by using the formula 26.5.16 of Abramowitz and Segun.

$$g(k) = I_\varepsilon(p,q) - I_\varepsilon(p+k,q+1) = K * (1 - (p+q)/q \varepsilon + (1-\varepsilon)R(k,\varepsilon)),$$

where $R(k,\varepsilon)$ is a sum of $k-1$ finite positive terms.

Note that as k increases by one, a new positive term is added to $R(k,\varepsilon)$ without changing existing terms. Since $R(k,\varepsilon)$ becomes larger as k increases, ε needs to be larger to make $g(k) > 0$. That is, as k increases, α increases.

By choosing ε close enough to 1, $g(k)$ can be negative for any $k \in \mathbb{Z}^+$. For such an $\varepsilon < 1$, (p,q) gives a higher confidence factor than $(p+k,q+1)$.

So, a range of ε , where (p,q) gives a higher confidence factor, is always nonempty for any $k \in \mathbb{Z}^+$.

By Lemma 4.4, $1 - I_\varepsilon(p+k,q+1)$ increases as k increases. So, the range of ε , where $(p+k,q+1)$ gives a higher confidence factor, is also always nonempty for any $k \in \mathbb{Z}^+$ if $q/(p+q) \geq b/m$.

b) In the proof of a) we showed that

$$h(k) = h(1) = I_\varepsilon(p,q) - I_\varepsilon(p+1,q+1) = K * (1 - (p+q)/q \varepsilon),$$

where $K = \varepsilon^p(1-\varepsilon)^q \Gamma(p+q) / \Gamma(p+1)\Gamma(q)$.

From Peizer and Pratt (1968), we get

$$I_c(p, q+1) - I_c(p, q) = \varepsilon^p (1-\varepsilon)^q \Gamma(p+q) / \Gamma(p) \Gamma(q+1).$$

By substituting $I_c(p+1, q+(k-1))$ for $I_c(p+1, q+k)$ using the above formula, we get a general formula for $h(k)$.

$$h(k) = I_c(p, q) - I_c(p+1, q+k) = K * (1 - (p+q)/q \varepsilon - (1-\varepsilon)L(k, \varepsilon)),$$

where $L(k, \varepsilon)$ is a sum of $k-1$ finite positive terms.

Note that as k increases by one, a new positive term is added to $L(k, \varepsilon)$ without changing existing terms. Since $L(k, \varepsilon)$ becomes larger as k increases, $h(k) < 0$ for smaller ε . So, the interval of ε , where (p, q) gives a higher confidence factor, is extended as k increases.

For $k=2$,

$$h(2) = I_c(p, q) - I_c(p+1, q+2) = K * (1 - (p+q)/q \varepsilon - ((p+q+1)(p+q) / (q+1)q) (1-\varepsilon)\varepsilon).$$

If $p/q = 1/2$, and p, q are large enough, then for $\varepsilon > 1/3$, $h(2) < 0$. As p/q increases, $(p+q)/q$ increases. Hence, β decreases.

□

By Lemma 4.6 we know that there is no (p, q) which gives the worst possible confidence factor, δ , over all ε , $b/m \leq \varepsilon < 1$, unless $q/(p+q) < b/m$. Since it is not practical to search for all consistent Beta priors and compare them to find a (p, q) giving the highest δ for each possible ε , we may consider a simple suboptimal solution strategy that gives a

near-optimal solution. For example, fixing p at a maximum level first, and then finding a minimum q , given p , is one strategy. Fixing q at a minimum level, and then finding a maximum p , given q , can be another strategy.

Since a maximum possible p is a very large indefinite number, we would be better off to fix q at the minimum level, and then find a maximum possible p with the given q .

In the following we give our main result of this subsection based on the above strategy.

Theorem 4.7: For $p, q \in Z^+$ and $b/m \leq \epsilon < 1$, where $1 \leq b \leq m-1$, then

$$(p, q) \in S(b, m)$$

only if $q \geq 4m(1-\epsilon)(\epsilon-b/m)$, where ϵ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - b/m$.

Let q^* be the minimum q satisfying the above condition, and let p^* be the maximum consistent p given q^* . Then (p^*, q^*) is a consistent Beta prior which gives the worst possible confidence factor for at least a certain nonempty interval of ϵ .

After testing with m_2 samples obtaining b_2 misclassifications, then

$$\text{Prob}\{ \theta \geq \epsilon \} \leq \delta, \quad \text{where } 1 - I_\epsilon(p^*+b_2, q^*+m_2-b_2) \equiv \underline{\delta} \leq \delta.$$

That is, (p^*, q^*) gives a lower bound for δ .

Proof: Theorem 4.7 follows from Lemma 4.3, 4.4, 4.5 and 4.6.

□

Consider an example of a training sample of size $m = 40$, and $b = 8$ misclassifications found by the pruned tree. Further suppose that we now take a sample of size $m_2 = 20$ and get two ($b_2 = 2$) misclassifications using the pruned tree.

By Theorem 4.7 we need to find the minimum q^* first. Since $b/m = 8/40 = 0.2$,

$\epsilon = .8189$ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - 0.2$.

Hence, $q \geq 4 \cdot 40(1-.8189)(.8189-.2) = 17.933$, and $q^* = 18$.

We now need to find p^* given $q^* = 18$. For $p = 2$, and $\epsilon = 0.8189$,

$$\begin{aligned} \text{Prob}\{ \theta \geq \epsilon \} &= 1 - I_c(2, 18) = (1-.8189)^{18}(1+18 \cdot .8189) = \\ &6.911 \times 10^{-13} > \exp(-2(\epsilon-b/m)^2 m) = \exp(-2(0.8189-.2)^2 \cdot 40) = \\ &4.919 \times 10^{-14}. \end{aligned}$$

So, $(p, q) = (2, 18)$ is not a consistent Beta prior. Hence, by Lemma 4.4, $p^* = 1$ given $q^* = 18$, and for $\epsilon = 0.2$,

$$\text{Prob}\{ \theta \geq 0.2 \} \leq \delta,$$

$$\text{where } 1 - I_c(p^*+b_2, q^*+m_2-b_2) = 1 - I_{.2}(1+2, 18+20-2) = \underline{\delta} \leq \delta.$$

Evaluating the left term gives

$$\text{Prob}\{ \theta \geq 0.2 \} \leq \delta, \text{ where } 1 - I_{.2}(3, 36) = .0113 = \underline{\delta} \leq \delta.$$

So the probability that the error of the pruned tree is greater than or equal to 0.2 is less than δ , where δ is bounded below by 0.0113.

For small ϵ , we may improve the above lower bound by finding a consistent Beta prior for larger p values. For example, if $p=2$, the smallest consistent q is 20. In this case,

$$\underline{\delta} = 1 - I_{\epsilon}(p^*+b_2, q^*+m_2-b_2) = 1 - I_{.2}(2+2, 20+20-2) = .0244.$$

By continuing the search, we can improve the lower bound for smaller ϵ . For larger ϵ , however, such as $\epsilon \geq 3/4$, $(p, q) = (2, 20)$ does not improve the bound.

For $\epsilon = 0.8$, $\text{Prob}(\theta \geq 0.8) \leq \delta$, where $\underline{\delta} = 1 - I_{.8}(3, 36) = .0000 \leq \delta$.

As we have seen in the above illustration, the range of ϵ , where (p, q) gives higher confidence factor, depends on many parameters, such as the increase in q by the unit increase in p (i.e., "k"), p , q , p/q ratio, b , m , b_2 , and m_2 , etc.

An upper bound of δ will be given in the next subsection.

4.2.3 A General Error Bound

In this subsection we develop a general bound on

$$\text{Prob}(\theta \geq \epsilon)$$

that requires no assumption on the prior distribution or on the domain of interest. This bound also ignores any information that may have been obtained during training. With

a pruned decision tree, training information is not often useful since we implicitly assume that there are many chance occurrences on the training sample. So the bounds we now develop are appropriate especially when the decision tree is pruned or when the training domain is different from the testing domain.

Suppose there are b misclassifications out of m independent examples using the pruned decision tree. Here b and m correspond to b_2 and m_2 of previous subsections, respectively.

Theorem 4.8: (Bound on the Posterior Estimate of the Confidence Parameter)

For any $b/m \leq \epsilon$,

$$\text{Prob}\{ \theta \geq \epsilon \} \leq \exp(-2(\epsilon - b/m)^2 m).$$

Proof: Theorem 4.8 follows from Lemma 4.2.

□

A direct application of Theorem 4.8 gives the following.

Corollary 4.9: (Confidence Estimate for an Error Range)

For any $b/m \leq \epsilon$ and $0 \leq \epsilon' \leq b/m$,

$$\text{Prob}\{ \epsilon' \leq \theta \leq \epsilon \} \geq 1 - \exp(-2(b/m - \epsilon)^2 m) - \exp(-2(b/m - \epsilon')^2 m).$$

Instead of Hoeffding's inequality, we may use more improved, but a more complicated bound from Johnson and Kotz (1969).

Lemma 4.10: (Johnson and Kotz 1969)

Let x_1, x_2, \dots, x_n be independent random variables with $0 \leq x_i \leq 1$ and $E[x_i] = \mu$, for $i = 1, 2, \dots, n$,

$$\bar{x} = (\sum_{i=1}^n x_i) / n.$$

Then,

$$\text{Prob}(\bar{x} - \mu \geq c) < \exp(-2nc^2 - (4/3)nc^4)$$

or

$$\text{Prob}(\bar{x} - \mu \geq c) < \exp(-2nc^2/2\mu(1-\mu) - (4/9)nc^4).$$

The following result can be directly derived from Lemma 4.10 and Theorem 4.8.

Theorem 4.11: (Improved Bounds on the Posterior Estimate of the Confidence Parameter)

For any $b/m \leq \epsilon$,

$$\text{Prob}(\theta \geq \epsilon) < \exp(-2(\epsilon-b/m)^2m - (4/3)(\epsilon-b/m)^4m)$$

or

$$\text{Prob}(\theta \geq \epsilon) < \exp(-(\epsilon-b/m)^2m/(2\theta(1-\theta)) - (4/9)(\epsilon-b/m)^4m).$$

We can apply the above results to estimate the error of any pruned decision trees.

Suppose we have 16 test examples and two of them are misclassified in the pruned decision tree. Under the assumption of a uniform prior, Theorem 4.0 gives

$$\text{Prob}\{ \theta \geq 0.2 \mid 2, 16 \} = \sum_{k=0}^2 C_k^{17} (0.2)^k (1-0.2)^{17-k} = 0.3096.$$

and

$$\text{Prob}\{ \theta \geq 0.3 \mid 2, 16 \} = \sum_{k=0}^2 C_k^{17} (0.3)^k (1-0.3)^{17-k} = 0.07739.$$

The probability that the error of the learned concept with pruning is greater than 20% and 30% is .3096 and 0.07739, respectively.

A general bound with no assumption on the prior distribution is given by Theorem 4.8 and Theorem 4.11. Theorem 4.8 gives

$$\text{Prob}\{ \theta \geq 0.2 \} \leq \exp(-2(0.2-2/16)^2 16) = .83527$$

and

$$\text{Prob}\{ \theta \geq 0.3 \} \leq \exp(-2(0.3-2/16)^2 16) = .37531.$$

Theorem 4.11 gives

$$\text{Prob}\{ \theta \geq 0.2 \} < \exp(-2(0.2-2/16)^2 16 - (4/3)(0.2-2/16)^4 16) = .8347$$

and

$$\text{Prob}\{ \theta \geq 0.3 \} < \exp(-2(0.3-2/16)^2 16 - (4/3)(0.3-2/16)^4 16) = .3678.$$

4.3 An Application

In Section 4.2 we give a lower bound for the worst possible confidence factor, δ , and a general upper bound for δ . In this section we combine these results to obtain a range where δ exists.

Suppose a training sample of size 20 is used in building a decision tree, and then the decision tree is pruned by a pruning technique, and gives 6 misclassifications. Further suppose the pruned decision tree is tested by 16 independent examples, and two misclassifications are obtained.

Since $b/m = 6/20 = 0.3$, $\epsilon = .8568$ is the root of the equation $-2(1-\epsilon)\ln(1-\epsilon) = \epsilon - 0.3$.

Hence, $q \geq 4*20(1-.8568)(.8568-.3) = 6.42$, and by Theorem 4.7, $q^* = 7$.

We now need to find p^* given $q^* = 7$. For $p = 2$, and $\epsilon = 0.8568$,

$$\begin{aligned} \text{Prob}\{ \theta \geq \epsilon \} &= 1 - I_c(2,7) = (1-.8568)^7(1+7*.8568) = \\ 0.0000084 &> \exp(-2(\epsilon-b/m)^2m) = \exp(-2(0.8568-.3)^2*20) = \\ .0000041. \end{aligned}$$

So, $(p,q) = (2,7)$ is not consistent Beta prior. Hence by Lemma 4.4, $p^* = 1$ given $q^* = 7$, and for $\epsilon = 0.3$,

$$\text{Prob}\{ \theta \geq 0.3 \} \leq \delta,$$

where $1 - I_c(p^*+b_2, q^*+m_2-b_2) = \underline{\delta} = 1 - I_{.3}(1+2, 7+16-2) = .0157 \leq \delta$.

We now give an upper bound for δ . Since Theorem 4.8 and 4.11 can be applied in any situation with independent random variables, x_i , these two theorems can give an upperbound for δ .

By Theorem 4.8,

$$\text{Prob}\{ \theta \geq 0.3 \} \leq \exp(-2(0.3-2/16)^2 16) = .37531.$$

and by Theorem 4.11,

$$\text{Prob}\{ \theta \geq 0.3 \} < \exp(-2(0.3-2/16)^2 16) - (4/3)(0.3-2/16)^4 16 = .3678.$$

So a range of δ is obtained, $.0157 \leq \delta \leq .3678$.

That is, the worst possible probability that the learned concept with pruning has an error greater than 30% could be as high as .3678, but not less than .0157. In this case, since ϵ is small, we can increase the lower bound by finding consistent Beta priors for higher p .

CHAPTER 5 AN INVESTIGATION ON THE CONDITIONS OF PRUNING

5.1 Introduction

Pruning leads to concept simplification. When does pruning lead to better predictions? In this chapter we develop conditions under which pruning is necessary to obtain better prediction accuracy. The analysis of this chapter implicitly assumes that there may be some hidden attributes (Spangler, et al. 1989, called this case the case of "inconclusive data"). That is, the current attribute set does not entirely determine its classification. Schaffer (1991) gives a result on the conditions under which overfitting (eg., an unpruned tree) does not decrease predictive accuracy. We give a different view of overfitting and generalize his result. We then apply this result to our specific pruning situation. Here we consider prediction accuracy of true class as the measure of the merit of pruning. In Section 5.2 we give and analyze the fundamental situation where pruning occurs. Following Schaffer, in Section 5.3 we perform a Bayesian analysis for samples of size three to find the conditions, under which pruning increases prediction accuracy as well as yielding concept simplicity. Finally in Section

5.4 we give a generalization of the results in Section 5.3 for the larger sample sets.

5.2 Fundamental Situation of Pruning

In an empirical study, Quinlan (1987a) found that pruning increases the accuracy of the learned concept. Mingers' (1989) empirical study showed that pruning improves the accuracy by 20% to 25%. Schaffer (1991) theoretically investigated pruning for very small sample sizes. We first summarize his results in a decision tree perspective.

Consider the simplest case of a decision tree, where a tree is a binary tree having only one node. We assume that there is no measurement noise in the instance space. We define measurement noise as the error occurring when we measure the values of attributes and classifications. We also assume an equally likely sampling distribution with replacement.

Schaffer's Fundamental Situation:

There are 4 possible decision trees with at most one node. The values of the attribute are 0 and 1. P and N denote class labels.

tree #1



tree #2



tree #3



tree #4



Suppose the true probabilities of class P for 0 and 1 are p_0 and p_1 . Then the errors of each tree are, respectively,

$$e_1 = (1 - p_0 + p_1) / 2 \quad \text{for tree \#1,}$$

$$e_2 = (p_0 + 1 - p_1) / 2 \quad \text{for tree \#2,}$$

$$e_3 = (1 - p_0 + 1 - p_1) / 2 \quad \text{for tree \#3, and}$$

$$e_4 = (p_0 + p_1) / 2 \quad \text{for tree \#4.}$$

Without loss of generality, consider a situation where we have to decide whether to prune tree #1 or not. By comparing the errors of each pruned and unpruned decision tree we can conclude the following. Note that e_1 is greater than e_3 when $p_1 > 0.5$, and that e_1 is greater than e_4 when $p_0 < 0.5$.

1. If $p_0 < 0.5$ and $p_1 > 0.5$ then pruning tree #1 increases its prediction accuracy since e_1 is greater than e_3 and e_4 . In this case, pruning increases the prediction accuracy by removing a less reliable branch. This branch reflects chance occurrences rather than representing a true underlying relationship.

2. If $p_0 < 0.5$ and $p_1 < 0.5$ then the prediction accuracy depends on the label of the pruned tree. For example, if the label of the pruned tree is N for the unpruned tree #1, then pruning increases the prediction accuracy. Otherwise pruning decreases the prediction accuracy.

3. If $p_0 > 0.5$ and $p_1 > 0.5$ then the prediction accuracy depends on the label of the pruned tree. For example, if the label of the pruned tree is P for the unpruned tree #1, then pruning increases the prediction accuracy.

4. If $p_0 > 0.5$ and $p_1 < 0.5$ then the unpruned tree #1 has a larger prediction accuracy regardless of the label of the pruned tree since e_1 is less than e_3 and e_4 . In this case, the branch reflects the true underlying relationship.

As we see in the above result, pruning may increase the prediction accuracy by removing branches constructed by the examples occurring by chance (eg., the case of $p_0 < 0.5$ and $p_1 > 0.5$). Note that the decision will be reversed if we have to decide whether to prune tree #2 or not. The above analysis, however, does not use any information from the training sample or any prior observations. In the next subsection, we summarize certain basic results using a Bayesian analysis.

5.3 A Bayesian Analysis on the Conditions Where Pruning is Useful

Schaffer (1991) gives a result for the above example using Bayesian methods. Consider a training sample of size three.

Sample #1: { (0,P), (0,P), (1,N) }

In this situation we have a choice between a simple hypothesis (tree #3) having error less than 50% over the training sample and a statistically less reliable, but better fitting complex hypothesis (tree #1). Schaffer calls this case an equivocal case. He investigates the conditions for

equivocal cases where the best fitting unpruned decision tree leads to a decrease in the prediction performance over the best fitting pruned decision tree.

We begin with a precise definition of an equivocal case.

Definition 5.0: (An equivocal case) Let n be the size of the training sample. Let q be the number of examples of the larger class in the sample. If q is less than n and strictly greater than $n/2$, then a $q:n-q$ partitioned sample of size n is called an equivocal case.

Let S_p be the strategy of choosing the best fitting pruned tree in an equivocal case and S_u be the strategy of choosing the best fitting unpruned tree.

5.3.1 A Bayesian Analysis

In the following, we restrict our attention to cases involving samples of size three. Suppose the prior joint distribution for (p_0, p_1) is uniform on the unit square. With this assumption and the assumption of a uniform distribution over the instance space, a Bayesian analysis gives the average prediction accuracy of the decision trees chosen by S_p and S_u for equivocal training sets. Details of the calculation process are as follows.

Since S_p and S_u ignore the order of sampling, there are just four possible equivocal training sets of size three.

$$S_1 = \{ (0,P), (0,P), (1,N) \},$$

$$S_2 = \{ (0,N), (0,N), (1,P) \},$$

$$S_3 = \{ (0,P), (1,N), (1,N) \}, \text{ and}$$

$$S_4 = \{ (0,N), (1,P), (1,P) \}.$$

$$\text{Let } k_1 = P\{(0,P)\}^2 P\{(1,N)\},$$

$$k_2 = P\{(0,N)\}^2 P\{(1,P)\},$$

$$k_3 = P\{(1,N)\}^2 P\{(0,P)\} \text{ and}$$

$$k_4 = P\{(1,P)\}^2 P\{(0,N)\}.$$

$$\text{Here } P\{(0,P)\} = p_0 / 2,$$

$$P\{(0,N)\} = (1-p_0) / 2,$$

$$P\{(1,P)\} = p_1 / 2 \text{ and}$$

$$P\{(1,N)\} = (1-p_1) / 2$$

since we assumed $P\{(0,.)\} = .5$. Then the probabilities of each sample over the space of equivocal training sets are

$$P(S_i) = k_i / K \text{ for } i = 1, 2, 3, 4, \text{ where } K = k_1 + k_2 + k_3 + k_4.$$

Let $A(T_i)$ be the accuracy of tree #i, where $i = 1, 2, 3, 4$, as measured by the probability of its predicting the true classification. Then

$$A(T_1) = (p_0 + (1-p_1))/2,$$

$$A(T_2) = ((1-p_0) + p_1)/2,$$

$$A(T_3) = (p_0 + p_1)/2 \text{ and}$$

$$A(T_4) = (1-(p_0 + p_1))/2.$$

Let $A(S_p)$ and $A(S_u)$ be the average accuracy of S_p and S_u . Since S_u chooses T_1 given S_1 or S_3 and T_2 given S_2 or S_4 ,

$$A(S_u) = (P(S_1) + P(S_3)) * A(T_1) + (P(S_2) + P(S_4)) * A(T_2).$$

Similarly, since S_p chooses T_3 given S_1 or S_4 and T_4 given S_2 or S_3 ,

$$A(S_p) = (P(S_1) + P(S_4)) * A(T_3) + (P(S_2) + P(S_3)) * A(T_4).$$

Then the difference of expected prediction accuracy is

$$\begin{aligned} E[A(S_p) - A(S_u)] &= \int_0^1 \int_0^1 (A(S_p) - A(S_u)) f(p_0, p_1) dp_0 dp_1 \\ &= (1/128) \int_0^1 \int_0^1 ((p_0)^2(1-p_1) - (1-p_0)^2(p_1))(2p_1-1) + ((p_1)^2(1-p_0) - (1-p_1)^2(p_0))(2p_0-1) dp_0 dp_1 \\ &= -10/288 < 0 \end{aligned}$$

since $f(p_0, p_1) = 1$ over the unit square.

This analysis shows that S_u yields decision trees with a higher average prediction accuracy than S_p (Schaffer 1991). That is, under the conditions described, pruning does not increase the prediction accuracy.

As we see in the above analysis, the result depends on the assumed prior distribution for p_0 and p_1 .

5.3.2 Disguised Bayesian Analysis Without Noise

In this subsection we assume that values for p_0 and p_1 are fixed, though unknown to us. We may calculate the performance of decision trees selected by the two strategies for each equivocal observation sequence and average these performance figures, weighted by the chance of various observation sequences arising under the assumed values of p_0 and p_1 . Schaffer (1991) called this type of analysis a "Disguised" Bayesian Analysis.

Here we use $A(S_p)$ and $A(S_u)$ as defined in the Subsection 5.3.1. We calculate $A(S_p)$ and $A(S_u)$ for each assumed values of p_0 and p_1 .

We say that S_p is preferable if $A(S_p)$ is greater than $A(S_u)$, and vice versa. By performing calculations for all possible pairs of p_0 and p_1 values, we get the following results.

S_p is preferable for points near the $p_0 = p_1$ line in the $p_0 > 0.5$ and $p_1 > 0.5$, or $p_0 < 0.5$ and $p_1 < 0.5$ regions. But the acceptable range grows as the parameters approach 0 or 1. The S_p preferred regions are much smaller than those in which S_u is preferable (Schaffer 1991). Approximate calculation shows that the S_u area is 65% of the total region.

Note that Schaffer does not consider the quality of the decision (i.e., the degree to which the prediction accuracies for the two strategies differ at a given (p_0, p_1) point). He also does not consider the distribution of p_0 and p_1 .

5.3.3 Disguised Bayesian Analysis With Noise

For more realistic learning environments, we may consider the possibility of erroneous observations and misclassifications.

Let e_d be the level of noise in the attributes (i.e., the description noise), the probability that the true value of an attribute is changed to some other value. Let e_c be the level of noise in the classification (i.e., the classification noise), the probability that the true classification is

changed to some other classification. Then the probabilities of making the four observations change as follows. Note that each term in the following equations comes from the four possible combinations of noise, i.e., the correct observation, the observation with classification noise only, the observation with description noise only, and the observation with description and classification noise.

$$P\{(0,P)\} = [(1-e_d)(1-e_c)p_0 + (1-e_d)e_c(1-p_0) + e_d(1-e_c)p_1 + e_de_c(1-p_1)]/2.$$

$$P\{(0,N)\} = [(1-e_d)(1-e_c)(1-p_0) + (1-e_d)e_cp_0 + e_d(1-e_c)(1-p_1) + e_de_cp_1]/2.$$

$$P\{(1,P)\} = [(1-e_d)(1-e_c)p_1 + (1-e_d)e_c(1-p_1) + e_d(1-e_c)p_0 + e_de_c(1-p_0)]/2.$$

$$P\{(1,N)\} = [(1-e_d)(1-e_c)(1-p_1) + (1-e_d)e_cp_1 + e_d(1-e_c)(1-p_0) + e_de_cp_0]/2.$$

Using a similar analysis procedure as in the previous subsection, Schaffer showed the following:

- 1) The classification noise has a nearly negligible effect on the relative merit of S_p and S_u .
- 2) The description noise has a strong effect, however, even at moderate levels (Schaffer 1991).

However, Schaffer does not investigate the case of $e_c > .5$ or $e_d > .5$. We have extended the analysis to the case of $e_c > .5$ or $e_d > .5$. The analysis has been done by using larger errors in Schaffer's formula. The following was found.

3) If the classification noise is greater than 0.5, then it has a strong effect on the relative merit of S_p and S_u . The S_p preferred region is close to 65% regardless of the level of classification noise.

4) If the description noise is greater than 0.5, then the S_p preferred region is close to 100% regardless of the level of description noise.

This result is consistent with Quinlan's (1986) empirical results at lower noise levels. Quinlan's result shows that the description noise has a strong effect at lower noise levels. However, the classification noise is more dangerous for higher noise levels (the case of $e_c > .7$).

5.4 A Generalization on the Conditions Where Pruning is Useful

In this section we generalize Schaffer's results for larger training sets. Note that pruning techniques are often applied to statistically unreliable branches. This observation leads to the following conjecture. Our conjecture is that the pruning decision (i.e., determining the relative merit of S_p over S_u) depends on the number of examples on each branch and on the ratio of the number of examples of the larger class to the size of the training set. We call this ratio the "skewness of the training set", and we begin with a precise definition of the skewness.

Definition 5.1: (The Skewness of the Training Set) Let n be the size of the training sample. Let q be the number of examples of the larger class in the sample. Then the ratio q/n is called the skewness of the training set.

If the skewness of the training set is large, then the branch of the smaller class is based on a small number of examples and it may be considered as chance occurrences rather than true underlying relationship. By removing these chance occurrences, pruning may improve the prediction accuracy over the instance space.

Consider sample sets S_1 , S_2 , S_3 , and S_4 given in Subsection 5.3.1.

$$S_1 = \{ (0,P), (0,P), (1,N) \},$$

$$S_2 = \{ (0,N), (0,N), (1,P) \},$$

$$S_3 = \{ (0,P), (1,N), (1,N) \}, \text{ and}$$

$$S_4 = \{ (0,N), (1,P), (1,P) \}.$$

The number of examples of the larger class is two for each training set and the sample size is three. In this case, the skewness of the training set is $2/3 = 0.667$.

In the following we assume that there exists a consistent decision tree, of given attributes, for the training set.

5.4.1 Varying the Skewness of the Training Set

In this subsection we increase the skewness of the training sets to investigate the effect of an unbalanced

sampling. Consider training sets of size four. We assume there is no noise. Since S_p and S_u ignore the order of sampling, there are just four equivocal training sets of size four. They are;

$$S_1 = \{ (0,P), (0,P), (0,P), (1,N) \},$$

$$S_2 = \{ (0,N), (0,N), (0,N), (1,P) \},$$

$$S_3 = \{ (0,P), (1,N), (1,N), (1,N) \}, \text{ and}$$

$$S_4 = \{ (0,N), (1,P), (1,P), (1,P) \}.$$

In this case the skewness of each training set is $3/4 = 0.750$.

In the case of samples of size five, there are eight possible equivocal training sets. Four of them are 4:1 partitions (i.e., four samples are in one class and only one is in the other class), and the remaining four are 3:2 partitions. If we consider only the 4:1 partitions, then the skewness is $4/5 = 0.800$. We refer to this case as "pure skewness". If we consider all possible equivocal training sets of size five, then the skewness can be calculated in at least two different ways. One is the simple average of the pure skewnesses. The other is a weighted average of skewnesses based on the number of permutations of each training set. We refer to these cases as "composite skewness". Thus, for training sets of size five we have

$$1) \text{ simple average} = (4/5 + 3/5) / 2 = .700, \text{ and}$$

$$2) \text{ weighted average} =$$

$$((C_1^5 * 4/5 + (C_1^4 + C_2^4) * 3/5) / (C_1^5 + C_1^4 + C_2^4)) = .667.$$

Here C_k^m is the number of ways of sampling k out of m things. The number of permutations of each training set is calculated as follows. Here we must consider the order of sampling. Suppose we have a $q:r$ partitioned training set, where q and r are positive integers. We know that counting the number of equivalent training sets is considered as a two step process.

Step 1. Choose t places out of $q+1$ places

Step 2. Spread r indistinguishable examples to t distinguishable places with an onto function.

Since t varies from 1 to r , the number of permutations is

$$\sum_{t=1}^r C_{t-1}^{r-1} * C_t^{q+1}.$$

Let Sample #2 = { (0,P), (0,P), (0,P), (1,N), (1,N) },

Sample #3 = { (0,P), (0,P), (1,N), (1,N), (0,P) }, and

Sample #4 = { (0,P), (0,P), (1,N), (0,P), (1,N) }.

All of the above are equivalent training sets if we ignore the order of sampling.

For above case, $q = 3$ and $r = 2$. So, the total number of ordered samples corresponding to this equivocal training set is $C_1^4 + C_2^4 = 10$.

In this subsection we consider "pure skewness" only. We will give a detailed analysis for the "composite skewness" case in the next subsection.

Change k_1 , k_2 , k_3 and k_4 as required. For example, if a training set S_1 consists of r of the (0,P) examples and 1 of the (1,N) example, then

$$k_1 = P\{(0,P)\}^2 P\{(1,N)\}^1.$$

Using a similar analysis procedure as given in Subsection 5.3.2, with the new k_1 , k_2 , k_3 and k_4 values, we determine the following (See Table 5.1).

1) As we increase the simple skewness ratio, the relative merit of S_p over S_u is increased in a log-like pattern. In other words, the relative merit increases rapidly for smaller values of the skewness ratio and increases slowly for larger values.

2) However, the least upper bound value of the S_p preferred region does not exceed 50%.

Table 5.1: Skewness vs. Size of S_p preferred region

Size of training set	Skewness of sample	Portion of S_p region

3	.667	.352 (Schaffer's case)
4	.750	.414
5	.800	.445 ({4:1} cases only)
6	.833	.464 ({5:1} cases only)
11	.909	.490 ({10:1} cases only)
very large	~.999	.500

Below Lemma 5.2 characterizes the limiting value of the S_p preferred region.

Lemma 5.2: The following are true.

- 1) If $p_0 < 0.5$ and $p_1 > 0.5$, then S_u is preferred.
- 2) If $p_0 > 0.5$ and $p_1 < 0.5$, then S_u is preferred.
- 3) If the skewness becomes close to 1, and if $p_0 > 0.5$ and $p_1 > 0.5$, then S_p is preferred.
- 4) If the skewness becomes close to 1, and if $p_0 < 0.5$ and $p_1 < 0.5$, then S_p is preferred.

Proof: 1) Let k be the number of examples represented by the larger class and let K be the sum of the probabilities of observing each training set defined in 5.3.1. Then

$$\begin{aligned}
 & K(A(S_u) - A(S_p)) \\
 = & ((p_0/2)^k((1-p_1)/2) + ((1-p_1)/2)^k(p_0/2))(p_0+1-p_1)/2 \\
 & + (((1-p_0)/2)^k(p_1/2) + (p_1/2)^k((1-p_0)/2))(1-p_0+p_1)/2 \\
 & - ((p_0/2)^k((1-p_1)/2) + (p_1/2)^k((1-p_0)/2))(p_0+p_1)/2 \\
 & - (((1-p_0)/2)^k(p_1/2) + ((1-p_1)/2)^k(p_0/2))(1-(p_0+p_1)/2). \\
 = & - ((2p_1-1)/2)((p_0/2)^k(1-p_1)/2 - ((1-p_0)/2)^k(p_1/2)) \\
 & - ((2p_0-1)/2)((p_1/2)^k(1-p_0)/2 - ((1-p_1)/2)^k(p_0/2)).
 \end{aligned}$$

Suppose $p_0 < 0.5$ and $p_1 > 0.5$. Then

$$\begin{aligned}
 & (p_0/2)^k((1-p_1)/2) < ((1-p_0)/2)^k(p_1/2) \text{ and} \\
 & (p_1/2)^k((1-p_0)/2) > ((1-p_1)/2)^k(p_0/2).
 \end{aligned}$$

So, $A(S_u) - A(S_p) > 0$.

2) Similar reasoning gives this result.

3) Let $f_1(k) = (p_0/2)^k((1-p_1)/2) - ((1-p_0)/2)^k(p_1/2)$ and $f_2(k) = (p_1/2)^k((1-p_0)/2) - ((1-p_1)/2)^k(p_0/2)$. Then $f_1(k) \geq 0$ if $k \geq \ln((1-p_1)/p_1) / \ln((1-p_0)/p_0)$ and

$f_2(k) \geq 0$ if $k \geq \ln((1-p_0)/p_0) / \ln((1-p_1)/p_1)$.

Let $k^* = \max (\ln((1-p_1)/p_1) / \ln((1-p_0)/p_0), \ln((1-p_0)/p_0) / \ln((1-p_1)/p_1))$. Then for any $p_0 > 0.5$ and $p_1 > 0.5$, we can find a k such that $k > k^*$. So, for such a $k > k^*$, $A(S_u) - A(S_p) < 0$. Hence, if the skewness becomes close to 1, then S_p is preferred.

4) Similar reasoning gives this result.

□

5.4.2 The Effect of a Larger Training Set and Noise

In 5.4.1 we increased the skewness of the training set. In order to give a comprehensive view on the effect of pruning on a larger training set, we consider both types of sampling strategies, i.e., unordered sampling and ordered sampling. These correspond to simple average composite skewness and weighted average composite skewness discussed in the previous subsection, respectively.

5.4.2.1 Simple average composite skewness

First we consider the unordered sampling case. In this case, when we increase the size of the training set, several different partitioning schemes are possible. For example, if the training set size is six, the following cases are possible.

1) 6:0 case: all six samples are positive examples, or all six examples are negative examples.

2) 5:1 case: five examples are in the same class (positive or negative) and the remaining one example is in the other class.

3) 4:2 case: four examples are in the same class (positive or negative) and the remaining two examples are in the other class.

4) 3:3 case: three examples are in the positive class and three examples are in the negative class.

For case 1), the sample strongly supports a simple hypothesis. For case 4), the given sample strongly supports a complex hypothesis since any simple hypothesis cannot explain more than 50% of the sample. Cases 3) and 4) are equivocal cases. We analyse the average performance of each hypothesis for these cases.

We use a similar analysis procedure as that used in Subsection 5.3.2. The number of equivocal training sets, the probabilities of observing each training set (i.e., k_1 , k_2 , etc), and the calculation of expected accuracy (i.e., $A(S_p)$ and $A(S_u)$) are modified appropriately. Since the ways of partitioning the sample are different depending on the sample size, the calculation formulae are not uniform for different training set sizes. However, they have a similar form and logic, we explain the procedure by giving an example for a sample of size six.

If we ignore the order of sampling, there are just eight equivocal training sets of size six. They are

$$\begin{aligned}
S_1 &= \{ (0,P), (0,P), (0,P), (0,P), (0,P), (1,N) \}, \\
S_2 &= \{ (0,N), (0,N), (0,N), (0,N), (0,N), (1,P) \}, \\
S_3 &= \{ (0,P), (1,N), (1,N), (1,N), (1,N), (1,N) \}, \\
S_4 &= \{ (0,N), (1,P), (1,P), (1,P), (1,P), (1,P) \}, \\
S_5 &= \{ (0,P), (0,P), (0,P), (0,P), (1,N), (1,N) \}, \\
S_6 &= \{ (0,N), (0,N), (0,N), (0,N), (1,P), (1,P) \}, \\
S_7 &= \{ (0,P), (0,P), (1,N), (1,N), (1,N), (1,N) \}, \text{ and} \\
S_8 &= \{ (0,N), (0,N), (1,P), (1,P), (1,P), (1,P) \}.
\end{aligned}$$

Then the probabilities of observing the training sets are, respectively,

$$\begin{aligned}
k_1 &= P\{(0,P)\}^5 P\{(1,N)\}^1, \\
k_2 &= P\{(0,N)\}^5 P\{(1,P)\}^1, \\
k_3 &= P\{(1,N)\}^5 P\{(0,P)\}^1, \\
k_4 &= P\{(1,P)\}^5 P\{(0,N)\}^1, \\
k_5 &= P\{(0,P)\}^4 P\{(1,N)\}^2, \\
k_6 &= P\{(0,N)\}^4 P\{(1,P)\}^2, \\
k_7 &= P\{(1,N)\}^4 P\{(0,P)\}^2, \text{ and} \\
k_8 &= P\{(1,P)\}^4 P\{(0,N)\}^2.
\end{aligned}$$

$P\{(0,P)\}$, $P\{(0,N)\}$, $P\{(1,P)\}$ and $P\{(1,N)\}$ are defined in the same way as in Subsection 5.3.3.

Then the probabilities of each sample over the space of all equivocal training sets are

$$P(S_i) = k_i / K \quad \text{for } i = 1, 2, 3, 4, 5, 6, 7, 8$$

where $K = k_1 + k_2 + k_3 + k_4 + k_5 + k_6 + k_7 + k_8$. The prediction accuracy of tree #i, $A(T_i)$, where $i = 1, 2, 3, 4$, is defined as

in Subsection 5.3.1. Since S_u chooses T_1 given S_1 or S_3 or S_5 or S_7 and T_2 given S_2 or S_4 or S_6 or S_8 ,

$$A(S_u) = (P(S_1) + P(S_3) + P(S_5) + P(S_7)) * A(T_1) + (P(S_2) + P(S_4) + P(S_6) + P(S_8)) * A(T_2).$$

Similarly, since S_p chooses T_3 given S_1 or S_4 or S_5 or S_8 and T_2 given S_2 or S_3 or S_6 or S_7 ,

$$A(S_p) = (P(S_1) + P(S_4) + P(S_5) + P(S_8)) * A(T_1) + (P(S_2) + P(S_3) + P(S_6) + P(S_7)) * A(T_2).$$

By performing calculations for all possible pairs of p_0 and p_1 values, we get the results shown in Table 5.2.

As we increase the size of the training sets, the skewness fluctuates between 0.700 and 0.750. We can calculate the asymptotic value of the skewness of the training set as follows. First consider even-numbered sample size. Let $2n$ be the size of the sample, where n is a whole number. Then

$$\begin{aligned} \text{skewness} &= 1/(n-1) * ((n+1)/2n + (n+2)/2n + \dots \\ &\quad + (2n-1)/2n) = 1/(n-1) * (3/4)(n-1) = 0.750. \end{aligned}$$

Now consider a sample set of size $2n-1$. Then

$$\begin{aligned} \text{skewness} &= 1/(n-1) * (n/(2n-1) + (n+1)/(2n-1) + \dots \\ &\quad + (2n-2)/(2n-1)) = 1/(n-1) * (n-1)(3n-2)/(4n-2) \\ &= (3n-2)/(4n-2). \end{aligned}$$

By taking the limit to n , we get the asymptotic value of skewness equal to 0.750.

Table 5.2: Sample size vs. S_p region for unordered sample

Size of training set	Skewness of set	Portion of S_p region*	e_d^* **
3	.667	.352	.34
4	.750	.414	.26
5	.700	.401	.30
6	.750	.436	.26
7	.714	.431	.28
8	.750	.455	.25
9	.722	.450	.27
10	.750	.465	.25
very large	.750	.500	.00

*: S_p preferred region has been calculated approximately by taking 10,000 points in the (p_0, p_1) plane.

**: e_d^* is the amount of description noise where the S_p preferred region is approximately equal to the S_u preferred region. Here we set $e_c = 0$.

As we see in the Table 5.2, the effect of a larger training set on the size of the S_p region is not monotone. As we increase the size of a training set from four samples to five, the relative merit of the pruned decision tree decreases. This is opposite to the case of increasing the sample size from three to four. (In this case, as we increase the size of a training set from three to four, the relative merit of the pruned decision tree increases). However, we can easily see that for odd or even numbered sequences (e.g., the 3,5,7,9 odd sequence or the 4,6,8,10 even sequence), the S_p ,

preferred regions are monotonically increasing in size. So we can say that the effect of larger training set is not monotone, and the relative merit is influenced by the skewness of the training set as well as the size itself. That is, for the same sample size, higher skewness gives a higher percentage of the S_p preferred region. If both the sample size and skewness increase, then the S_p preferred region increases.

These observations are generalized in the following.

Lemma 5.3: Let n be an odd number such that $n \geq 3$. If the sample size increases from n to $n+1$, then the S_p preferred region does not decrease.

Proof: For any $q:r$ partitioning of n , a $(q+1):r$ partitioning of $n+1$ exists. Let f_n be the difference of the prediction accuracy between S_p and S_u for the sample of size n . Suppose $p_0 > 0.5$ and $p_1 > 0.5$. Then f_{n+1} is obtained by multiplying the positive terms of f_n by $p_0/2$ or $p_1/2$ and the negative terms of f_n by $(1-p_0)/2$ or $(1-p_1)/2$. So, if f_n is positive, then f_{n+1} is also positive since $p_0/2 > (1-p_0)/2$, and $p_1/2 > (1-p_1)/2$, and sum of multipliers are equal to 0.5 in both cases. Hence, if $A(S_p) > A(S_u)$ for n , then this inequality also holds for $n+1$. This implies that the S_p preferred region does not decrease. For the case of $p_0 < 0.5$ and $p_1 < 0.5$, by the formula of f_n given in Proof of Lemma 5.2, we multiply the positive terms by

a corresponding larger number and negative terms by a smaller number. Hence, if $A(S_p) > A(S_u)$ for n , then this inequality also holds for $n+1$.

□

We have also investigated the influence of description noise and classification noise. We have not seen any notable evidence that the effect of classification noise is influenced by the change of the size of the training sets. However, the effect of description noise becomes stronger as the size of the training set increases. That is, the amount of description noise where the S_p preferred region is approximately equal to the S_u preferred region is reduced as the size of the training set increases. The range of description noise for which S_p has relative merit over S_u is extended to relatively lower noise levels. We also see that the relative importance of description noise to classification noise does not change as we change the skewness and the size of the training sets.

These observations are generalized below.

Lemma 5.4: For the unordered sampling case, the limiting value for the size of the S_p preferred region is .500 as the sample size goes to infinity.

Proof: Let n be an even-numbered sample size. Then the difference of expected prediction accuracy, f_n , is

$$\begin{aligned}
 f_n &= K(A(S_p) - A(S_u)) \\
 &= ((2p_1-1)/2)((p_0/2)^{n-1}(1-p_1)/2 - ((1-p_0)/2)^{n-1}(p_1/2)) \\
 &+ ((2p_0-1)/2)((p_1/2)^{n-1}(1-p_0)/2 - ((1-p_1)/2)^{n-1}(p_0/2)) \\
 &+ ((2p_1-1)/2)((p_0/2)^{n-2}((1-p_1)/2)^2 - ((1-p_0)/2)^{n-2}(p_1/2)^2) \\
 &+ ((2p_0-1)/2)((p_1/2)^{n-2}((1-p_0)/2)^2 - ((1-p_1)/2)^{n-2}(p_0/2)^2) \\
 &+ \dots \\
 &+ ((2p_1-1)/2)((p_0/2)^{n/2+k}((1-p_1)/2)^{n/2-k} - ((1-p_0)/2)^{n/2+k}(p_1/2)^{n/2-k}) \\
 &+ ((2p_0-1)/2)((p_1/2)^{n/2+k}((1-p_0)/2)^{n/2-k} - ((1-p_1)/2)^{n/2+k}(p_0/2)^{n/2-k}) \\
 &+ \dots \\
 &+ ((2p_1-1)/2)((p_0/2)^{n/2+1}((1-p_1)/2)^{n/2-1} - ((1-p_0)/2)^{n/2+1}(p_1/2)^{n/2-1}) \\
 &+ ((2p_0-1)/2)((p_1/2)^{n/2+1}((1-p_0)/2)^{n/2-1} - ((1-p_1)/2)^{n/2+1}(p_0/2)^{n/2-1})
 \end{aligned}$$

For an odd-numbered sample size n , the terms $n/2+1$ and $n/2-1$ in the above would be changed to $(n+1)/2$ and $(n-1)/2$, respectively. We focus on the case where n is even.

Let $g(k) =$

$$\begin{aligned}
 &((2p_1-1)/2)((p_0/2)^{n/2+k}((1-p_1)/2)^{n/2-k} - ((1-p_0)/2)^{n/2+k}(p_1/2)^{n/2-k}) \\
 &+ ((2p_0-1)/2)((p_1/2)^{n/2+k}((1-p_0)/2)^{n/2-k} - ((1-p_1)/2)^{n/2+k}(p_0/2)^{n/2-k}),
 \end{aligned}$$

where $k = 1, 2, \dots, n/2-1$.

By rearranging the terms we get

$$\begin{aligned}
 g(k) &= (p_1/2)^{n/2-k}((1-p_0)/2)^{n/2-k} * \\
 &(((2p_0-1)/2)(p_1/2)^{2k} - ((2p_1-1)/2)((1-p_0)/2)^{2k}) \\
 &+ (p_0/2)^{n/2-k}((1-p_1)/2)^{n/2-k} * \\
 &(((2p_1-1)/2)(p_0/2)^{2k} - ((2p_0-1)/2)((1-p_1)/2)^{2k}).
 \end{aligned}$$

Suppose $p_0 > 0.5$ and $p_1 > 0.5$. Without loss of generality, further suppose $p_1 > p_0$. Then, for any k ,

$$((2p_1-1)/2)(p_0/2)^{2k} - ((2p_0-1)/2)((1-p_1)/2)^{2k} > 0.$$

If $k \geq \ln((2p_0-1)/(2p_1-1)) / 2 \ln(p_0/(1-p_1))$, then

$$((2p_0-1)/2)(p_1/2)^{2k} - ((2p_1-1)/2)((1-p_0)/2)^{2k} \geq 0.$$

Let k^* be the smallest integer not smaller than

$$\ln((2p_0-1)/(2p_1-1)) / 2 \ln(p_0/(1-p_1)).$$

Then $g(k) \geq 0$ for all $k \geq k^*$.

Since $g(k+1)$ is obtained by multiplying positive terms of $g(k)$ by $p_1/(1-p_0)$ or $p_0/(1-p_1)$, and by multiplying negative terms by the inverse of previous factors (they are less than 1),

$$g(k+1) > g(k) \text{ for all } k.$$

Note that following algebraic fact holds. If $A - B = 0$ and $\alpha > 1$, then $\alpha A - (1/\alpha)B = -((1/\alpha)A - \alpha B)$. Since $g(k^*) \geq 0$, by the above fact, $g(k^*+1) + g(k^*-1) \geq 0$.

Hence, $g(k^*+t) + g(k^*-t) \geq 0$ for all $t < k^*$.

Since k^* is finite for a fixed (p_0, p_1) point, f_n can be made positive by choosing n sufficiently large. That is, for sufficiently large n ,

$$A(S_p) - A(S_u) > 0 \text{ if } p_0 > 0.5 \text{ and } p_1 > 0.5.$$

$$\text{Similarly, } A(S_p) - A(S_u) > 0 \text{ if } p_0 < 0.5 \text{ and } p_1 < 0.5.$$

When $p_0 < 0.5$ and $p_1 > 0.5$, or when $p_0 > 0.5$ and $p_1 < 0.5$, $A(S_p) - A(S_u) < 0$ by the same reasoning given in the proof of Lemma 5.2, the limiting value for the size of the S_p preferred region is .500.

□

Corollary 5.5: Let n be a sufficiently large number. If the sample size increases from n to $n+2$, then the S_p preferred region does not decrease.

Proof: Let n be the sample size. Without loss of generality we assume n is an even number. Then the difference of expected prediction accuracy, f_n , is defined as in the proof of Lemma 5.4.

Define $g(k)$ and k^* the same way as in the proof of Lemma 5.4, where $k = 1, 2, \dots, n/2-1$.

Let $g(k) = g_1(k) + g_2(k)$,

where $g_1(k) = (p_1/2)^{n/2-k} ((1-p_0)/2)^{n/2-k} *$

$((2p_0-1)/2) (p_1/2)^{2k} - ((2p_1-1)/2) ((1-p_0)/2)^{2k}$, and

$g_2(k) = (p_0/2)^{n/2-k} ((1-p_1)/2)^{n/2-k} *$

$((2p_1-1)/2) (p_0/2)^{2k} - ((2p_0-1)/2) ((1-p_1)/2)^{2k}$.

Suppose $p_0 > 0.5$ and $p_1 > 0.5$. Without loss of generality, further suppose $p_1 > p_0$. Then, for any k ,

$((2p_1-1)/2) (p_0/2)^{2k} - ((2p_0-1)/2) ((1-p_1)/2)^{2k} > 0$.

Suppose $f_n > 0$. f_n can be written as follows.

$f_n = g(1) + g(2) + \dots + g(k) + \dots + g(n/2-1)$.

Write f_{n+2} as follows.

$f_{n+2} = h(1) + h(2) + \dots + h(k) + \dots + h(n/2)$,

where $h(k) = (p_1/2)^{(n+2)/2-k} ((1-p_0)/2)^{(n+2)/2-k} *$

$((2p_0-1)/2) (p_1/2)^{2k} - ((2p_1-1)/2) ((1-p_0)/2)^{2k}$

$+ (p_0/2)^{(n+2)/2-k} ((1-p_1)/2)^{(n+2)/2-k} *$

$((2p_1-1)/2) (p_0/2)^{2k} - ((2p_0-1)/2) ((1-p_1)/2)^{2k}$.

$$\begin{aligned} \text{Then } f_{n+2} = & h(n/2) + (p_1/2)((1-p_0)/2) * (g_1(1) + \dots + g_1(n/2-1)) \\ & + (p_0/2)((1-p_1)/2) * (g_2(1) + \dots + g_2(n/2-1)). \end{aligned}$$

For sufficiently large n , if $f_n > 0$,

then $g_1(1) + \dots + g_1(n/2-1) > 0$ must hold.

Since $h(n/2) > 0$, and all multipliers in above equation are positive, $f_{n+2} > 0$ follows.

For $p_0 < 0.5$ and $p_1 < 0.5$, $f_{n+2} > 0$ by a similar reasoning.

□

5.4.2.2 Weighted average skewness

Here we consider the number of permutations of each training set, and give weights to the probabilities, k_i , and skewnesses by those numbers. The analysis procedure is similar to the "simple average" case. However, each k_i is multiplied by the number of permutations. By the formula given in Subsection 5.4.1, we get the number of permutations. For example, for a training set of size six, the number of permutations are obtained as follows. For a 5:1 partitioned training sets (i.e., five examples are in one class and one example is in the other class), the number of permutations is $C_1^5 = 6$. And for a 4:2 partitioned training sets, the number of permutations is $C_1^5 + C_2^5 = 15$. So, we get:

$$k_1 = 6 * P\{(0,P)\}^5 P\{(1,N)\}^1,$$

$$k_2 = 6 * P\{(0,N)\}^5 P\{(1,P)\}^1,$$

$$k_3 = 6 * P\{(1,N)\}^5 P\{(0,P)\}^1,$$

$$k_4 = 6 * P\{(1,P)\}^5 P\{(0,N)\}^1,$$

$$k_5 = 15 * P\{(0,P)\}^4 P\{(1,N)\}^2,$$

$$k_6 = 15 * P\{(0,N)\}^4 P\{(1,P)\}^2,$$

$$k_7 = 15 * P\{(1,N)\}^4 P\{(0,P)\}^2, \text{ and}$$

$$k_8 = 15 * P\{(1,P)\}^4 P\{(0,N)\}^2.$$

Then the skewness is

$$= 6/21 * 5/6 + 15/21 * 4/6 = .714.$$

By performing calculations for all possible pairs of p_0 and p_1 values, we get the following results shown in Table 5.3.

Table 5.3: Sample size vs. S_p region for ordered sample

Size of training set	Skewness of set	Portion of S_p region [*]	e_d ^{**}
3	.667	.352	.34
4	.750	.414	.26
5	.667	.380	.33
6	.714	.422	.29
7	.651	.394	.35
8	.685	.428	.31
9	.635	.401	.36
10	.662	.429	.33
11	.623	.411	.37
12	.645	.434	.35
very large		.500	.00

*: S_p preferred region has been calculated approximately by taking 10,000 points in the (p_0, p_1) plane.

**: e_d is the amount of description noise where the S_p preferred region is approximately equal to the S_u preferred region. Here we set $e_c = 0$.

In Table 5.3 we see the same effect of a larger training set as found in Table 5.2. That is, we can easily see that for even or odd numbered sequences, the S_p preferred regions are monotonically increasing. However, the increase is

relatively slow compared to the former "simple average skewness" case.

Even though the S_p preferred regions are monotonically increasing, the corresponding e_d^* is increasing. This is opposite to the case of "simple average skewness". This can be explained by the declining skewness since e_d^* decreases as the skewness increases, as shown in Tables 5.1 and 5.2.

Lemma 5.6: Let $q:r$ and $(q+1:r-1)$ be two partitions of training sample. Then the number of permutations of $(q+1:r-1)$ partitioned training sample is $r/(q+1)$ times of the number of permutations of $(q:r)$ partitioned training sample.

Proof: Note that the number of permutations for $q:r$ partitioned training sample is $\sum_{t=1}^r C_{t-1}^{r-1} * C_t^{q+1}$.

Equate the number of permutations for $q:r$ and $(q+1:r-1)$ partitioned training sample by using an unknown multiple x .

$$\sum_{t=1}^{r-1} C_{t-1}^{r-2} * C_t^{q+2} = x \sum_{t=1}^r C_{t-1}^{r-1} * C_t^{q+1}.$$

We can rewrite the above equation as follows.

$$\sum_{t=0}^{r-2} C_t^{r-2} * C_{t+1}^{q+2} = x \sum_{t=0}^{r-1} C_t^{r-1} * C_{t+1}^{q+1}.$$

By the identity 3.20 of Gould (1972) which is

$$\sum_{k=0}^n C_k^n * C_{k+r}^x = C_{n+r}^{n+x},$$

the above equation reduces to

$$C_{r-1}^{r+q} = x C_r^{r+q}.$$

Hence, $x = r / (q+1)$ follows.

□

Lemma 5.7: For the ordered sampling case, the limiting value for the size of the S_p preferred region is .500 as the sample size n goes to infinity.

Proof: The ratio of the number of permutations for $k+1$ (i.e., a partition where the difference between q and r is $2(k+1)$) to the number of permutations for k is

$$(n-2k) / (n+2k+2) \quad \text{for even number } n, \text{ or}$$

$$(n-2k+1) / (n+2k+1) \quad \text{for odd number } n$$

by Lemma 5.6. This ratio is getting larger as n increases for any fixed k . Since k^* is finite by the proof of Lemma 5.4, the sum of the number of permutations for all $k < k^*$ can be exceeded by the sum of the number of permutations for all $k > k^*$ for sufficiently large n . Since the expected prediction accuracies are multiplied by the number of permutations, the reasoning in the proof of Lemma 5.4 holds for this case also.

□

5.4.3 Discussion

Ideally, pruning should lead to concept simplification with better predictions or at least a small loss of prediction accuracy. In this chapter we have developed conditions under which pruning is necessary to obtain better prediction accuracy. We have analyzed this problem in three directions.

- 1) Increasing skewness.
- 2) Increasing sample size.

3) Increasing noise levels.

All three have a positive effect on the relative merit of pruning. In particular, for higher skewness and/or larger training sets, the S_p preferred region approaches 50% when there is no noise of any kind. That is, a pruned tree has almost an equal level of prediction accuracy as that of unpruned tree. The situation becomes more favorable to the pruned tree when description noise exists.

Now we give our main result in Theorem 5.8.

Theorem 5.8: Under the assumption of a uniform distribution, as the skewness increases and/or sample size increases, the merit of pruning increases, and the limiting value for the size of the S_p preferred region is 50%.

We also considered the degree to which prediction accuracies for the two strategies differ at a given (p_0, p_1) point. The amount of difference in quality of these strategies depends on the distributions for p_0 and p_1 . If we assume a uniform distribution, and take 10,000 points at equal intervals from the (p_0, p_1) unit square, we get the average of the differences of the prediction accuracies, shown in Table 5.4. The results are based on the "weighted average" cases described in Subsection 5.4.2.2.

Table 5.4: Average difference of prediction accuracy

Sample size	Skewness	S_p region	Difference*
3	.667	.353	.073
4	.750	.414	.041
10	.662	.429	.030
11	.623	.411	.037
12	.645	.434	.028

*: This is the average difference of the prediction accuracy between S_p and S_u for a fixed (p_0, p_1) point.

As we see in Table 5.4, the loss of prediction accuracy for the pruned tree is less than 4% for sample sizes greater than ten. This small amount of loss in prediction accuracy as a result of pruning is often an acceptable trade-off for producing a simple concept.

CHAPTER 6

SUMMARY AND FUTURE RESEARCH

Empirical results have shown that pruning can improve the accuracy of an induced decision tree. Pruning also leads to more concise rules. In Chapter 3 we provide a pruning algorithm based on the rank of a decision tree. A bound on the error due to pruning by the rank of a decision tree is determined under the assumptions of an equally likely distribution over the instance space and a deterministic tree labelling rule. This bound is then used with recent results in learning theory to determine a sample size sufficient for PAC identification of decision trees with pruning. We also discuss other pruning rules and their effects on the error due to pruning. With a nondeterministic tree labelling rule we show that the upperbound of the average pruning error is less than or equal to 0.5 under an equally likely distribution.

In Chapter 3 we provide a bound on the training sample size to guarantee PAC learning of a decision tree with pruning. In a realistic learning environment it is often not possible to obtain a large enough sample. For those cases, we provide several methods for a posterior evaluation of the accuracy of a pruned decision tree in Chapter 4. We give a

method which estimates a lower bound for the worst possible confidence factor, δ , by using a Beta prior. Also, we give a more detailed view of the meaning of this lower bound, and suggest a way to improve this lower bound.

In Chapter 5 we develop conditions under which pruning is necessary for better prediction accuracy as well as for concept simplification. We give an analysis of the reason why pruning is necessary in realistic learning situations.

We generalize Schaffer's (1991) results for larger training sets. A Bayesian analysis shows that the average prediction accuracy of the pruned tree increases, and the effect of description noise becomes stronger as the size of the training set increases. For very large training sets, the pruned tree has the prediction accuracy equal to that of the unpruned tree.

Future work will be needed to determine the pruning error under more general assumptions on the distribution over the instance space. Also, Theorem 3.23 can be tightened if an a priori estimate, k , of the rank of the induced decision tree can be determined. If so, $\mu_{n,r}$ can be replaced by $\mu_{k,r}$.

Here we take the rank of a tree as a conciseness measure of a decision tree. Future work will be needed to assess the effect of pruning under other conciseness criteria.

Future work will be needed to find a more direct, and convenient way to find δ , in particular, a way to find an improved upperbound of δ .

Future work will be also needed to investigate the conditions, under which pruning is useful, for more complex situations, such as larger decision trees having more than one node and nonbinary trees.

REFERENCES

- Abramowitz, M. and Segun, I. (1968), Handbook of Mathematical Functions, Dover Publications, New York.
- Angluin, D. and Laird, P. (1988), "Learning From Noisy Examples," Machine Learning, 2, 343-370.
- Angluin, D. and Smith, C.H. (1983), "Inductive Inference: Theory and Methods," Computing Surveys, 15, 237-269.
- Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. (1987a), "Occam's Razor," Information Processing Letters, 24, 377-380.
- Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. (1987b), "Learnability and the Vapnik-Chervonenkis Dimension," Technical Report UCSC-CRL-87-20, University of California, Santa Cruz, CA.
- Boose, J. H. and Gaines, B.R. (1989), "Knowledge Acquisition for Knowledge-Based Systems: Notes on the State-of-the Art," Machine Learning, 4, 377-394.
- Boucheron, S. and Sallantin, J. (1988), "Learning in the Presence of Noise," Proceedings of the Third European Working Session on Learning, London, 25-35.
- Braun, H. and Chandler, J.S. (1987), "Predicting Stock Market Behavior Through Rule Induction: An Application of the Learning-from-Example Approach," Decision Sciences, 18, 415-429.
- Breiman, L., Friedman, J., Olshen, R. and Stone, C. (1984) Classification and Regression Trees, San Francisco: Wadsworth International.
- Buntine, W. (1989), "Learning Classification Rules using Bayes," Proceedings of the 6th International Conference on Machine Learning (pp.146-150), San Mateo, CA: Morgan Kaufmann.
- Carter, C. and Catlett, J. (1987), "Assessing Credit Card Applications Using Machine Learning," IEEE Expert, 2, 71-79.

- Chan, P.K. (1989), "Inductive Learning with BCT," Proceedings of the 6th International Workshop on Machine Learning (pp. 104-108), San Mateo, CA: Morgan Kaufmann.
- Chan, K. and Wong, A. (1990), "Performance Analysis of a Probabilistic Inductive Learning System," Proceedings of the 7th International Conference on Machine Learning (pp.16-23), San Mateo, CA: Morgan Kaufmann.
- Cheng, J., Fayyad, U.M., Irani, K.B. and Qian, Z. (1988), "Improved Decision Trees: A Generalized Version of ID3," Proceedings of the 5th International Conference on Machine Learning (pp. 100-106), San Mateo, CA: Morgan Kaufmann.
- Chou, P.A. (1991), "Optimal Partitioning for Classification and Regression Trees," IEEE Transactions on Pattern Analysis and Machine Intelligence, 13, 4, 340-354.
- Cohen, P. R. and Feigenbaum, E.A. (1982), The Handbook of Artificial Intelligence, Vol III, Reading, MA: Addison-Wesley.
- Clark, P. and Niblett, T. (1987), "Induction in Noisy Domains," Proceedings of the Second European Working Session on Learning (pp. 11-30), Bled, Yugoslavia: Sigma Press.
- Clark, P. and Niblett, T. (1989), "The CN2 Induction Algorithm," Machine Learning, 3, 261-283.
- Davis, R. (1982), "TEIRESIAS: Applications of Meta-Knowledge," In Knowledge Based Systems in Artificial Intelligence (pp. 227-408), R. Davis and D. Renat (Eds.), New York: McGraw-Hill.
- Ehrenfeucht, A. and Haussler, D. (1988), "Learning Decision Trees From Random Examples," Proceedings of the 1988 Workshop on Computational Learning Theory (pp. 182-194), San Mateo, CA: Morgan Kaufmann.
- Ehrenfeucht, A., Haussler, D., Kearns, M. and Valiant, L. (1989), "A General Lower Bound on the Number of Examples Needed for Learning," Information and Computation, 82, 3, 247-261.
- Fisher, D. H. and Schlimmer, J. C. (1988), "Concept Simplification and Prediction Accuracy," Proceedings of the 5th International Conference on Machine Learning (pp. 22-28), San Mateo, CA: Morgan Kaufmann.

- Garson, B. (1988), The Electronic Sweatshop: How Computers Are Transforming the Office of the Future Into the Factory of the Past, New York: Simon and Schuster.
- Gelfand, S.B., Ravishankar, C.S., and Delp, E.J. (1991), "An Iterative Growing and Pruning Algorithm for Classification Tree Design," IEEE Transactions on Pattern Analysis and Machine Intelligence, 13, 2, 163-174.
- Gould, H.W. (1972), Combinatorial Identities, Morgantown, WV: Morgantown Printing and Binding Co.
- Hausssler, D. (1988), "Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework," Artificial Intelligence, 36, 177-221.
- Hausssler D. (1989), "Learning Conjunctive Concepts in Structural Domains," Machine Learning, 4, 7-40
- Hausssler, D. (1990), "Decision Theoretic Generalization of the PAC Model for Neural Net and Other Learning Applications," Technical Report, UCSC-CRL-91-02, University of California, Santa Cruz.
- Hirsh, H. (1990), "Learning from Data with Bounded Inconsistency," Proceedings of the 7th International Conference on Machine Learning (pp.32-39), San Mateo, CA: Morgan Kaufmann.
- Johnson, D.E. (1983), "What Kind of Expert Should a System Be?," Journal of Medicine and Philosophy, 8, 77-97.
- Johnson, N. and Kotz, S. (1969), Discrete Distributions, Boston: Houghton Mifflin Co.
- Koehler, G.J. and Majthay, A. (1988), "Generalization of Quinlan's Induction Method," Department of Decision and Information Sciences, University of Florida, Unpublished Manuscript.
- Laird, P. (1987), Learning From Good Data and Bad. Doctoral Dissertation, Department of Computer Science, Yale University, New Haven, CT.
- Landell, L. (1990), "Induction as Optimization," IEEE Transactions on Systems, Man, and Cybernetics, 20, 2, 326-338.
- Marshall, R. (1986), Partitioning Methods for Classification and Decision Making in Medicine, Statistics in Medicine, 5, 517-526.

- Messier, W.F. and Hansen, J.V. (1988), "Inducing Rules for Expert Systems Development," Management Science, 34, 12, 1403-1415.
- Michalski, R. S. (1983), "A Theory and Methodology of Inductive Learning," Artificial Intelligence, 20, 111-161.
- Michalski, R.S. and Chilausky, C. (1980), "Learning by Being Told and Learning From Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis," International Journal of Policy Analysis and Information Systems, 4, 125-161.
- Mingers, J. (1986), "Expert Systems--Experiments with Rule Induction," Journal of the Operational Research Society, 37, 1031-1037.
- Mingers, J. (1987), "Expert Systems--Rule Induction with Statistical Data," Journal of the Operational Research Society, 38, 39-47.
- Mingers, J. (1989a), "An Empirical Comparison of Selection Measures for Decision Tree Induction," Machine Learning, 3, 319-342.
- Mingers, J. (1989b), "An Empirical Comparison of Pruning Methods for Decision Tree Induction," Machine Learning 4, 227-243.
- Mitchell, T.M. (1982), "Generalization as Search," Artificial Intelligence, 18, 203-226.
- Musen, M. A. (1989), "Automated Support for Building and Extending Expert Models," Machine Learning, 4, 347-375.
- Natarajan, B. K. (1991), Machine Learning: A Theoretical Approach, San Mateo, CA: Morgan Kaufmann.
- Niblett, T. and Bratko, I. (1986), "Learning Decision Rules in Noisy Domains," In M.A. Bramer (Ed.), Research and Development in Expert Systems III(pp. 25-34), Cambridge: Cambridge University Press.
- Niblett, T. (1987), "Constructing Decision Trees in Noisy Domains," Proceedings of the Second European Working Session on Learning (pp. 67-78), Bled, Yugoslavia: Sigma Press.
- Nunez, M. (1991), "The Use of Background Knowledge in Decision Tree Induction," Machine Learning, 6, 231-250.

- Parker, C.S. (1989), Management Information Systems: Strategy and Action, New York: McGraw-Hill.
- Peizer, D.B. and Pratt, J.W. (1968), "A Normal Approximation for Binomial, F, Beta and other Common Related Tail Probabilities I," Journal of the American Statistical Association, 63, 1416-1456.
- Quinlan, R. (1979), "Discovering Rules from Large Collection of Examples: A Case Study," In D. Michie(Ed.), Expert Systems in the Microelectronic Age (pp. 168-201). Edinburgh: Edinburgh University Press.
- Quinlan, R. (1983), "The Effect of Noise in Concept Learning," In R.S. Michalski, J. Carbonell, T. Mitchell(Eds.), Machine Learning: An Artificial Intelligence Approach. Vol. II, Chapter 6, Los Altos, CA: Morgan Kaufmann.
- Quinlan, R. (1986), "Induction of Decision Trees," Machine Learning, 1, 86-106.
- Quinlan, R. (1987a), "Simplifying Decision Trees," International Journal of Man-Machine Studies, 27, 221-234.
- Quinlan, R. (1987b), "Generating Production Rules from Decision Trees," Proceedings of the 10th International Joint Conference on Artificial Intelligence (pp. 304-307), Los Altos, CA: Morgan Kaufmann.
- Quinlan, R. (1990), "Decision Trees and Decisionmaking," IEEE Transactions on Systems, Man, and Cybernetics, 20, 2, 339-346.
- Quinlan, R. and Rivest, R.L. (1989), "Inferring Decision Trees Using the Minimum Description Length Principle," Information and Computation, 80, 227-248.
- Raiffa, H. and Schlaifer (1961), Applied Statistical Decision Theory, Cambridge, MA: Division of Research, Harvard Business School.
- Rivest, R. (1987), "Learning Decision Lists," Machine Learning, 2(3), 229-246.
- Schaffer, C. (1991), "When does Overfitting Decrease Prediction Accuracy in Induced Decision Trees and Rule Sets?" Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91, Porto, Portugal: Springer-Verlag.
- Shackelford, G. and Volper, D. (1988), "Learning k-DNF with Noise in the Attributes," Proceedings of the 1988

- Workshop on Computational Learning Theory (pp. 97-103), San Mateo, CA: Morgan Kaufmann.
- Shaw, M. J. and Gentry, J.A. (1988), "Using an Expert System with Inductive Learning to Evaluate Business Loans," Financial Management, 17, 45-56.
- Shaw, M. J. and Gentry, J.A. (1990), "Inductive Learning for Risk Classification," IEEE Expert, 5, 47-53.
- Shaw, M. J., Gentry, J.A. and Piramuthu, S. (1990), "Inductive Learning Methods for Knowledge-Based Decision Support: A Comparative Analysis," Computer Science in Economics and Management, 3, 147-165.
- Simon, H. (1983), "Why Should Machines Learn?" In R.S. Michalski, J. Carbonell, T. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach. Vol. I (pp. 25-37), Palo Alto, CA: Tioga.
- Simon, H. U. (1990), "On the Number of Examples and Stages Needed for Learning Decision Trees," Proceedings of the Third Annual Workshop on Computational Learning Theory (pp. 303-313), San Mateo, CA: Morgan Kaufmann.
- Spangler, S., Fayyad, U.M. and Uthurusamy, R. (1989), "Induction of Decision Trees from Inconclusive Data," Proceedings of the 6th International Conference on Machine Learning (pp.146-150), San Mateo, CA: Morgan Kaufmann.
- Tsai, L. and Koehler, G.J. (in Press), "The Accuracy of Concepts Learned from Induction," Decision Support System, forthcoming.
- Utgoff, P. (1989), "Incremental Induction of Decision Trees," Machine Learning, 4, 161-186.
- Utgoff, P. and Brodley, C. (1990), "An Incremental Method for Finding Multivariate Splits for Decision Trees," Proceedings of the 7th International Conference on Machine Learning (pp.58-65), San Mateo, CA: Morgan Kaufmann.
- Valiant, L.G. (1984), "A Theory of the Learnable," Communications of the ACM, 27(11), 1134-1142.
- Valiant, L.G. (1985), "Learning Disjunctions of Conjunctions," Proceedings of the 9th International Joint Conference on Artificial Intelligence (Vol. 1, pp. 560-566), Los Angeles, CA: Morgan Kaufmann.

- Van de Velde, W. (1990), "Incremental Induction of Topologically Minimal Decision Trees," Proceedings of the 7th International Conference on Machine Learning (pp.66-74), San Mateo, CA: Morgan Kaufmann.
- Vapnik, V.N. (1982), Estimation of Dependencies Based on Empirical Data, New York: Springer-Verlag.
- Weiss, S. M. and Galen, R.S., and P.V. Tadepalli (1987), "Optimizing the Predictive Value of Diagnostic Decision Rules," Proceedings of the Sixth National Conference on Artificial Intelligence (pp. 521-526), San Mateo, CA: Morgan Kaufmann.
- Weiss, S. M. and Kulikowski, C. A. (1991), Computer Systems that Learn, Los Altos, CA: Morgan Kaufmann.
- Winkler, R. L. (1972), Introduction to Bayesian Inference and Decision, New York: Holt, Rinehart and Winston.
- Wirth, J. (1988), "Experiments on the Costs and Benefits of Windowing in ID3," Proceedings of the 5th International Conference on Machine Learning (pp.87-99), San Mateo, CA: Morgan Kaufmann.
- Zhou, X.J. and Dillon, T.S. (1991), "A Statistical-Heuristic Feature Selection Criterion for Decision Tree Induction," IEEE Transactions on Pattern Analysis and Machine Intelligence, 13, 8, 834-841.


BIOGRAPHICAL SKETCH

Hyunsoo Kim was born on August 6, 1958 in Geumreung, Korea, and moved to Seoul, Korea, in 1967, where he lived until he came to the United States for advanced study in 1989. He graduated Munsung elementary school, Kangnam middle school, and Baemoon high school in 1971, 1974, and 1977, respectively. In March of 1977 he entered Seoul National University and graduated in February 1982 with a B.S. degree in nuclear engineering.

With a strong motivation to build a career in business and management he joined the Management Science graduate program in Korea Advanced Institute of Science and Technology (KAIST) in March 1983, and received his master's degree in Management Science with a management information systems (MIS) concentration in February 1985. From March 1985 to May 1988 he was with Data Communications Corp. of Korea, Seoul, Korea, where he had finished several pioneering research projects in MIS and software engineering including "Software Development and Maintenance Cost Estimation in Korea." From June 1988 to August 1989 he was with the Information Culture Center, Seoul, Korea, where he taught many courses in MIS.


In August 1989 he joined the Ph.D. program at the University of Florida. He has been involved in various research projects in expert systems and machine learning. He has written four papers for publication as a part of his dissertation.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.




Gary J. Koehler, Chair
Professor of Decision and Information
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.




Harold P. Benson
Professor of Decision and Information
Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Mark Pendergast
Assistant Professor of Decision and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



William Messer
Professor of Accounting

This dissertation was submitted to the Graduate Faculty of the Department of Decision and Information Sciences in the College of Business Administration and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1992

Dean, Graduate School